

# glossaries-extra.sty v1.34: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2018-07-29

## 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	39
1.3.1	Existence Checks	43
1.3.2	Document Definitions	51
1.3.3	Existing Glossary Style Modifications	57
1.3.4	Entry Formatting, Hyperlinks and Indexing	61
1.3.5	Entry Counting	98
1.3.6	Acronym Modifications	113
1.3.7	Indexing and Displaying Glossaries	115
1.4	Link Counting	152
1.5	Integration with glossaries-accsupp	154
1.6	Categories	167
1.7	Abbreviations	193
1.7.1	Abbreviation Styles Setup	214
1.7.2	Predefined Styles (Default Font)	217
1.7.3	Predefined Styles (Small Capitals)	234
1.7.4	Predefined Styles (Fake Small Capitals)	248
1.7.5	Predefined Styles (Emphasized)	262
1.7.6	Predefined Styles (User Parentheses Hook)	284
1.7.7	Predefined Styles (Hyphen)	293
1.7.8	Predefined Styles (No Short on First Use)	307
1.8	Using Entries in Headings	310
1.9	Multi-Lingual Support	329
1.10	glossaries-extra-bib2gls.sty	330
<b>2</b>	<b>Style Adjustments (glossaries-extra-stylemods.sty)</b>	<b>366</b>
2.1	Package Initialisation	366
2.2	List-Like Styles	367
2.3	Longtable Styles	370
2.4	Long Ragged Styles	372
2.5	Supertabular Styles	374
2.6	Super Ragged Styles	376
2.7	Inline Style	378
2.8	Tree Styles	378
2.9	Multicolumn Styles	396

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

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

## 1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/07/29 v1.34 (NLCT)]

Requires xkeyval to define package options.
3 \RequirePackage{xkeyval}

Requires etoolbox package.
4 \RequirePackage{etoolbox}

Has glossaries already been loaded?
5 \@ifpackageloaded{glossaries}
6 {%

Already loaded so pass any options to \setupglossaries. This means that the options that
can only be set when glossaries is loaded can't be used.
7   \newcommand{\glstr@dooption}[1]{\setupglossaries{#1}}%
8   \let\@glstr@declareoption\@gls@declareoption
9 }
10 {%

Not already loaded, so pass options to glossaries.
11   \newcommand{\glstr@dooption}[1]{%
12     \PassOptionsToPackage{#1}{glossaries}%
13   }%

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

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

```

undefaction

```

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

```

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

\glstr@record Does nothing by default.

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

```

\glstr@recordsee Does nothing by default.

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

```

\glstr@defaultnumberformat

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

```

\GlsXtrSetDefaultNumberFormat

```

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

```

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

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

```

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

```

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

```

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

```

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

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



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

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

Save the label in case it's needed.

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

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

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

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

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

Implement any pre-key settings.

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

Assign keys.

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

Implement any post-key settings.

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

Increment associated counter.

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

Check if noindex option has been used.

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

Check if thevalue has been set.

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

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

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

Save the entry counter.

```
147         \glsxtr@saveentrycounter
```

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

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

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

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

No need to escape special characters.

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

glslink@prekeys

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

lslink@postkeys

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

lossadd@prekeys

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

ossadd@postkeys

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

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

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

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

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

```

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

```

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

```

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

```

`r@recordcounter`

```

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

```

`p@recordcounter`

```

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

```

`op@recordcounter`

```

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

```

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

```

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

```

srtglossaryunit

```

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

```

tr@setup@record Initialise.

```

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

```

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

```

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

```

addloclistfield

```

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

```

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

```

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

```

Add a key to store the group heading.

```

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

```

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

```

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

```

`setting@alsoindex`

```

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

```

`rd@setting@only`

```

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

```

`ord@setting@off`

```

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

```

`record` Now define the record package option.

```

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

```

Don't record.

```

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

```

Only record (don't index).

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

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

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

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

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

Warn about using `\printglossary`:

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

Load glossaries-extra-bib2gls:

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

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

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

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

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

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

Need to provide conditional commands that are backward compatible:

if@glxtrdocdef

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

lsxtrdocdeftrue

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

sxtrdocdeffalse

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

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

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

ocdefrestricted

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

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

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

indexcrossrefs Automatically index cross references at the end of the document

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

Switch off since this can increase the build time.

```
339 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

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

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

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

iesExtraWarning Allow users to suppress warnings.

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

raWarningNoLine Allow users to suppress warnings.

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

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

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

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

postdot Shortcut for nopostdot=false

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

nopostdot Needs to redefine \@glxtr@defpostpunc

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

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

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



```

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

```

`\glsxtrabbrvtype` Glossary type for abbreviations.

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

`\bbreviationsdef` Set by abbreviations option.

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

`\bbreviationsdef`

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

Disable this command after it's been used.

```

444   \let\GlsXtrDefineAbbreviationShortcuts\relax
445 }

```

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

```

446 \newcommand*{\GlsXtrDefineAcShortcuts}{%

```

```

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

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

```

Disable this command after it's been used.

```

472 \let\GlsXtrDefineAcShortcuts\relax
473 }

```

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

```

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

```

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

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

```

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

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

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

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

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

523       \GlsXtrDefineAcShortcuts
524       \GlsXtrDefineAbbreviationShortcuts

```

```

525     \GlsXtrDefineOtherShortcuts
526 }%

```

```

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

```

Leave none and false as last option.

```

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

```

`lsxtr@doaccsupp`

```

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

```

`accsupp` If `accsupp`, load glossaries-accsupp package.

```

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

```

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

```

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

```

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

```

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

```

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

`lsxtr@redefstyles`

```

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

```

stylemods

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

glxtr@do@style

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

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

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

Defer actual style change:

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

Set this as the default style:

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

Set this style:

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

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

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

`ocationHyperlink`

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

The first two arguments are always control sequences.

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

`cationhyperlink`

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

`indexcounter`

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

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

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

Only increment if the current counter is `wrglossary`.

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

```

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

```

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

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

```

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

```

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

```

`sxtrwrglossmark` Does nothing by default.

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

```

`debug` Provide extra debug options.

```

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

```

Pass all other options to glossaries.

```

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

```

Process options.

```
653 \ProcessOptionsX

```



Load glossaries if not already loaded.

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

Load the glossaries-accsupp package if required.

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

Redefine \glspostdescription if required.

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

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

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

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

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

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

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

oindex@glossary

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

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

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

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

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

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

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

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

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

Define abbreviations glossaries if required.

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

Setup shortcuts if required.

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

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

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

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

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

Now define the user command:

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

```

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

```

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

```

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

```

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

```

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

```

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

```

726 \let\glxtr@saveentrycounter\@gl@saveentrycounter

```

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

```

727 \let\@gl@saveentrycounter\glxtr@indexonly@saveentrycounter

```

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

`sxtrdialecthook`

```

728 \newcommand*{\@glxtrdialecthook}{}

```

Set up record option if required.

```

729 \glxtr@setup@record

```

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

```

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

```

## 1.2 Extra Utilities

`unusedOrUndefined`

```

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

```

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

```

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

```

```

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

```

`\glsxtrifemptyglossary` `\glsxtrifemptyglossary{<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 `\gloclist@<type>`. If this hasn't been defined, the glossary doesn't exist. If it has been defined and is simply a comma, the glossary exists and is empty. (It's initialised to a comma.)

```

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

```

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

```

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

```

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

```

752 \newcommand*\glsxtrprovidestoragekey{%
753   \@ifstar\@sglsxtr@provide@storagekey\@glsxtr@provide@storagekey
754 }

```

`\@glsxtr@provide@storagekey` Unstarred version.

```

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

```

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

```

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

```

Provide the no-link command if not already defined.

```

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

```

`\provide@storagekey` Starred version.

```

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

```

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

`\GlsXtrFmtField`

```

790 \newcommand{\GlsXtrFmtField}{useri}

```

`\tDefaultOptions`

```

791 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

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

```

`\@glxtrfmt` Unstarred form.

```

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

```

`\s@glstrfmt` Starred form.

```

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

```

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

```

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

```

`\@glstrfmt` Actual inner working.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`\glstrentryfmt` No link or indexing.

```

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

```

```

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

```

@glstrententryfmt

```

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

```

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

```

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

```

trfieldlistgadd Similarly but uses \listcsgadd.

```

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

```

trfieldlisteadadd Similarly but uses \listcseadd.

```

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

```

trfieldlistxadd Similarly but uses \listcsxadd.

```

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

```

Now provide commands to iterate over these lists.

fieldddolistloop

```

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

```

ieldforlistloop

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

List element tests:

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

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

rfieldxifinlist Expands item.

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

lsxtrforcsvfield

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

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

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

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

lsxtrifhasfield Unstarred version adds grouping.

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

lsxtrifhasfield Starred version omits grouping.

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



```

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

```

`rIfFieldNonZero` Designed for numeric fields.

```

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

```

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

Designed for numeric fields.

```

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

```

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

Designed for numeric fields.

```

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

```

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

Just uses `\ifcsundef`.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`etfieldifexists`

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

**xGlsXtrSetField**

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

**eGlsXtrSetField**

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

**XtrIfFieldEqStr**

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

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

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

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

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

lsXtrForeignText

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

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

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

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

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

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

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

No tracked dialect found for the root language.

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

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

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

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

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

Does a caption hook exist for this?

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

No mapping. Try root language label instead.

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

No mapping. Try root language label instead.

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

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

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

nDialectWarning

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

```

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

```

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

```

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

```

`lossarypreamble`

```

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

```

lossarypreamble

```

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

```

## 1.3 Modifications to Commands Provided by glossaries

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

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

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

```

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

```

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

ewglossaryentry

```

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

```

ewglossaryentry Starred version.

```

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

```

```

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

```

`\newglossaryentry` Unstarred version.

```

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

```

The following is different from the base glossaries.sty:

```

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

```

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

```

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

```

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

`\ignoredglossary` Redefine to check for star.

```

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

```

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

```

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

```



```

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

```

ignoredglossary Starred form.

```

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

```

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

```

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

```

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

```

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

```

`ignoredglossary` Unstarred version.

```

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

```

```

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

```

`ignoredglossary` Starred form.

```

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

```

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

```

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

```

### 1.3.1 Existence Checks

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

```

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

```

1274 {%

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

1275 \edef\glslabel{\glsdetoklabel{#1}}%

1276 \glstrundefaction{Glossary entry '\glslabel'

1277 has not been defined}{You need to define a glossary entry before  
1278 you can reference it.}%

1279 }%

1280 }

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

1281 \renewcommand{\glsdoifnoexists}[2]{%

1282 \ifglentryexists{#1}{%

1283 \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'

1284 has already been defined}{}}{#2}%

1285 }

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

1286 \ifdef\glsdoifexistsordo

1287 {%

1288 \renewcommand{\glsdoifexistsordo}[3]{%

1289 \ifglentryexists{#1}{#2}%

1290 {%

1291 \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'

1292 has not been defined}{You need to define a glossary entry  
1293 before you can use it.}%

1294 #3%

1295 }%

1296 }%

1297 }

1298 {%

1299 \glstr@warnonexistsordo\glsdoifexistsordo

1300 \newcommand{\glsdoifexistsordo}[3]{%

1301 \ifglentryexists{#1}{#2}%

1302 {%

1303 \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'

1304 has not been defined}{You need to define a glossary entry  
1305 before you can use it.}%

1306 #3%

1307 }%

1308 }%

1309 }

arynoexistsordo Similarly for \doifglossarynoexistsordo.

1310 \ifdef\doifglossarynoexistsordo

1311 {%

1312 \renewcommand{\doifglossarynoexistsordo}[3]{%

```

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

```

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

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

```

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

```

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

```

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

```

```

1354 }

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

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

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

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

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

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

```

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

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

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

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

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

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

Add the cross-reference class definition to the hook.

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

Append to class list.

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

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

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

```

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

```

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

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

```

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

```

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

```

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

```

Add to the key mappings.

```

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

```

Set the default value.

```

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

```

Assign the field values.

```

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

```

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

```

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

```

Provide user-level commands to access the values.



```

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

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

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

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

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

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

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

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

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

`\glsxtrunusedformat`

```

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

```

### 1.3.2 Document Definitions

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

```

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

```

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

```

1546 \@gls@enablesavenonumberlist

```

```

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

```

Write all currently defined entries.

```

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

```

m@glossaryentry

```

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

```

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

```

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

```

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

```

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

```

Check for docdefs=restricted:

```

1584   \if@glsxtrdocdefrestricted

```

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

```

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

```

Disable document definitions.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

`rEnableOnTheFly`

```

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

```

`rEnableOnTheFly`

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

```

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

```

`sxtrstarflywarn`

```

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

```

`rEnableOnTheFly`

```

1650 \newcommand*{\@GlsXtrEnableOnTheFly}{%

```

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

These definitions are all assigned the category given by:

```

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

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

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

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

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

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

\@glstrpl

```

```

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

```

\Glsxtrpl

```

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

```

\@Glsxtrpl

```

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

```

\GlsXtrWarning

```

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

```

Disable commands after the glossary:

```

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

```



abledflycommand

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

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

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

### 1.3.3 Existing Glossary Style Modifications

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

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

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

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

etglossarystyle

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

Only set the current style if it exists.

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

In case we have an old version of glossaries:

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

```

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

```

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

```

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

```

Similarly for \glsdescwidth:

\glsdescwidth

```

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

```

and for \glspagelistwidth:

lspagelistwidth

```

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

```

aryentrynumbers Has the nonumberlist option been used?

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

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

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

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

ePreLocationTag

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

```

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

```

glsxtrpreloctag

```

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

```

glsxtrpreloctag

```

1855 \newcommand*{\@glsxtrpreloctag}{%

```

@glsxtr@pagetag

```

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

```

glsxtr@pagetag

```

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

```

lsxtrpostloctag

```

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

```

lsxtrpostloctag

```

1865 \newcommand*{\@glsxtrpostloctag}{%

```

lsxtr@preloctag

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

glxtr@doloctag

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

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

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

### 1.3.4 Entry Formatting, Hyperlinks and Indexing

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

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

`sxtrregularfont` Font used for regular entries.

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

`bbreviationfont` Font used for abbreviation entries.

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

\@Gls@ Save the original definition and redefine.

```

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

```

\@Glspl@ Save the original definition and redefine.

```

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

```

\@GLS@ Save the original definition and redefine.

```

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

```

\@GLSpl@ Save the original definition and redefine.

```

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

```

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

```

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

```

```

1957 \fi
1958 }%
1959 \glspostlinkhook
1960 }

```

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

```

1961 \renewcommand*{\@gls@@link}[3][\fi
1962 \glstr@record{#1}{#2}{glslink}%
1963 \glsdoifexistsordo{#2}%
1964 {%
1965 \let\do@gls@link@checkfirsthyper\relax
1966 \@gls@link[#1]{#2}{#3}%
1967 }%
1968 {%
1969 \glstextformat{#3}%
1970 }%
1971 \glspostlinkhook
1972 }

```

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

```

1973 \newcommand*{\glstrinitwrgloss}{%
1974 \glsifattribute{\glslabel}{wrgloss}{after}%
1975 {%
1976 \glstrinitwrglossbeforefalse
1977 }%
1978 {%
1979 \glstrinitwrglossbeforetrue
1980 }%
1981 }

```

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

```

1982 \newif\ifglstrinitwrglossbefore
1983 \glstrinitwrglossbeforetrue

```

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

```

1984 \define@choicekey{glslink}{wrgloss}%
1985 [\@glstr@wrglossval\@glstr@wrglossnr]%
1986 {before,after}%
1987 {%
1988 \ifcase\@glstr@wrglossnr\relax
1989 \glstrinitwrglossbeforetrue
1990 \or
1991 \glstrinitwrglossbeforefalse
1992 \fi
1993 }

```

```

1994 \define@key{glslink}{thevalue}{\def\@glstr@thevalue{#1}}

```



```

1995 \define@key{glslink}{theHvalue}{\def\@glstr@theHvalue{#1}}

tr@hyperoutside Define a hyperoutside key to determine whether \hyperlink should be outside \glstextformat.
1996 \define@boolkey{glslink}[glstr@]{hyperoutside}[true]{}
1997 \glstr@hyperoutsidettrue

ocal@textformat Provide a key to locally change the text format.
1998 \define@key{glslink}{textformat}{%
1999   \ifcsdef{#1}
2000   {%
2001     \letcs{\@glstr@local@textformat}{#1}%
2002   }%
2003   {%
2004     \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
2005   }%
2006 }

2007 \define@key{glslink}{prefix}{\def\glolinkprefix{#1}}

nithyperoutside Set the default if the hyperoutside is omitted.
2008 \newcommand*{\glstrinithyperoutside}{%
2009   \glsifattribute{glslabel}{hyperoutside}{false}%
2010   {%
2011     \glstr@hyperoutsidedefalse
2012   }%
2013   {%
2014     \glstr@hyperoutsidettrue
2015   }%
2016 }

r@inc@linkcount Does nothing by default.
2017 \newcommand*{\glstr@inc@linkcount}{}

slinkpresetkeys User hook performed immediately before options are set. Does nothing by default.
2018 \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.
2019 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
2020   \protected@edef\@glstr@tmp{#2}%
2021   \expandafter#1\expandafter{\@glstr@tmp}%
2022 }

\@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.
2023 \def\@gls@link[#1]#2#3{%

```

```

2024 \leavevmode
2025 \edef\glslabel{\glsdetoklabel{#2}}}%
2026 \def\@gls@link@opts{#1}%
2027 \let\@gls@link@label\glslabel
2028 \let\@gls@numberformat\@glsxtr@defaultnumberformat
2029 \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
2030 \edef\@gls@type{\csname glo@\glslabel @type\endcsname}%
2031 \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

  Save current value of \glslinkprefix:
2032 \let\@glsxtr@org@glslinkprefix\glslinkprefix

  Initialise \@glsxtr@local@textformat
2033 \let\@glsxtr@local@textformat\relax

  Initialise thevalue and theHvalue (v1.19).
2034 \def\@glsxtr@thevalue{}%
2035 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%

  Initialise when indexing should occur (new to v1.14).
2036 \glsxtrinitwrgloss

  Initialise whether \hyperlink should be outside \gls@textformat (new to v1.21).
2037 \glsxtrinithyperoutside

  Note that the default link options may override \glsxtrinitwrgloss.
2038 \@gls@setdefault@glslink@opts

  Increment link counter if enabled (new to v1.26).
2039 \glsxtr@inc@linkcount

  As the original definition.
2040 \do@gl:disablehyperinlist
2041 \do@gls@link@checkfirsthyper

  User hook before options are set (new to v1.26):
2042 \glslinkpresetkeys

  Set options.
2043 \setkeys{glslink}{#1}%

  User hook after options are set:
2044 \glslinkpostsetkeys

  Check thevalue and theHvalue before saving (v1.19).
2045 \ifdefempty{\@glsxtr@thevalue}%
2046 {%
2047   \@gls@saveentrycounter
2048 }%
2049 {%
2050   \let\thegl@sentrycounter\@glsxtr@thevalue
2051   \def\theHgl@sentrycounter{\@glsxtr@theHvalue}%
2052 }%
2053 \@gls@setsort{\glslabel}%

```

Check if the textformat key has been used.

```
2054 \ifx\@glstr@local@textformat\relax
```

Check textformat attribute (new to v1.21).

```
2055 \glshasattribute{\glslabel}{textformat}%
2056 {%
2057 \edef\@glstr@attrval{\glsetattribute{\glslabel}{textformat}}%
2058 \ifcsdef{\@glstr@attrval}%
2059 {%
2060 \letcs{\@glstr@textformat}{\@glstr@attrval}%
2061 }%
2062 {%
2063 \GlossariesExtraWarning{Unknown control sequence name
2064 '\@glstr@attrval' supplied in textformat attribute
2065 for entry '\glslabel'. Reverting to default \string\glstextformat}%
2066 \let\@glstr@textformat\glstextformat
2067 }%
2068 }%
2069 {%
2070 \let\@glstr@textformat\glstextformat
2071 }%
2072 \else
2073 \let\@glstr@textformat\@glstr@local@textformat
2074 \fi
```

Do write if it should occur before the link text:

```
2075 \ifglstr@nitr@wrglossbefore
2076 \do@wrglossary{#2}%
2077 \fi
```

Do the link text:

```
2078 \ifKV@glslink@hyper
2079 \ifglstr@hyperoutside
2080 \@glslink{\glolinkprefix\glslabel}{\@glstr@textformat{#3}}%
2081 \else
2082 \@glstr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
2083 \fi
2084 \else
2085 \ifglstr@hyperoutside
2086 \glndonohyperlink{\glolinkprefix\glslabel}{\@glstr@textformat{#3}}%
2087 \else
2088 \@glstr@textformat{\glndonohyperlink{\glolinkprefix\glslabel}{#3}}%
2089 \fi
2090 \fi
```

Do write if it should occur after the link text:

```
2091 \ifglstr@nitr@wrglossbefore
2092 \else
2093 \do@wrglossary{#2}%
2094 \fi
```

Restore original value of \glolinkprefix:

```
2095 \let\glolinkprefix\@glxtr@org@glolinkprefix
```

As the original definition:

```
2096 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2097 }
```

```
2098 \define@key{glossadd}{thevalue}{\def\@glxtr@thevalue{#1}}
```

```
2099 \define@key{glossadd}{theHvalue}{\def\@glxtr@theHvalue{#1}}
```

lsaddpresetkeys

```
2100 \newcommand*{\gladdpresetkeys}{}
```

saddpostsetkeys

```
2101 \newcommand*{\gladdpostsetkeys}{}
```

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

```
2102 \renewrobustcmd*{\gladd}[2][\]{%
2103   \glxtrifinmark
2104   {}}%
2105   {%
2106     \@gls@adjustmode
2107     \@glxtr@record{#1}{#2}{glossadd}%
2108     \glsdoifexists{#2}%
2109     {%
2110       \let\@glsnumberformat\@glxtr@defaultnumberformat
2111       \edef\@gls@counter{\csname glo@\glsetoklabel{#2}@counter\endcsname}%
2112       \def\@glxtr@thevalue{}%
2113       \def\@glxtr@theHvalue{\@glxtr@thevalue}%

```

Implement any default settings (before options are set)

```
2114   \gladdpresetkeys
2115   \setkeys{glossadd}{#1}%

```

Implement any default settings (after options are set)

```
2116   \gladdpostsetkeys
2117   \ifdefempty{\@glxtr@thevalue}%
2118   {%
2119     \@gls@saveentrycounter
2120   }%
2121   {%
2122     \let\theglentrycounter\@glxtr@thevalue
2123     \def\theHglentrycounter{\@glxtr@theHvalue}%
2124   }%

```

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

```
2125   \@gls@setsort{#2}%
2126   \@do@wrglossary{#2}%
2127 }
```

```

2128 }%
2129 }

```

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

```

2130 \newrobustcmd{\glsaddeach}[2] []{%
2131   \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%
2132 }

```

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

```

2133 \newcommand*{\@glsxtr@field@linkdefs}{%
2134   \let\glsxtrifwasfirstuse\@secondoftwo
2135   \let\glsifplural\@secondoftwo
2136   \let\glschapscase\@firstofthree
2137   \let\glsinsert\@empty
2138 }

```

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

`assignfieldfont`

```

2139 \newcommand*{\glsxtrassignfieldfont}[1]{%
2140   \ifglstryexists{#1}%
2141   {%
2142     \ifglshasshort{#1}%
2143     {%
2144       \glssetabbrvfmt{\glscategory{#1}}%
2145       \glsifregular{#1}%
2146       {\let\@gls@field@font\glsxtrregularfont}%
2147       {\let\@gls@field@font\@firstofone}%
2148     }%
2149     {%
2150       \glsifnotregular{#1}%
2151       {\let\@gls@field@font\@firstofone}%
2152       {\let\@gls@field@font\glsxtrregularfont}%
2153     }%
2154   }%
2155   {%
2156     \let\@gls@field@font\@gobble
2157   }%
2158 }

```

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

```

2159 \def\@glstext@#1#2[#3]{%
2160   \glsxtrassignfieldfont{#2}%
2161   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
2162 }

```

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

```

2163 \def\@GLStext@#1#2[#3]{%

```

```

2164 \glstrassignfieldfont{#2}%
2165 \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
2166 {\@gls@field@font{\Glsaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
2167 }

```

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

```

2168 \def\@Glstext@#1#2[#3]{%
2169 \glstrassignfieldfont{#2}%
2170 \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
2171 {\@gls@field@font{\Glsaccesstext{#2}#3}}%
2172 }

```

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

`ecknohyperfirst`

```

2173 \newcommand*\glstrchecknohyperfirst[1]{%
2174 \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
2175 }

```

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

```

2176 \def\@glsfirst@#1#2[#3]{%
2177 \glstrassignfieldfont{#2}%
2178 \gls@field@link
2179 [\let\glstrifwasfirstuse\@firstoftwo
2180 \glstrchecknohyperfirst{#2}%
2181 ]{#1}{#2}%
2182 {\@gls@field@font{\glsaccessfirst{#2}#3}}%
2183 }

```

Ensure that `\glsfirst` honours the `nohyperfirst` attribute.

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

```

2184 \def\@Glsfirst@#1#2[#3]{%
2185 \glstrassignfieldfont{#2}%
2186 \gls@field@link
2187 [\let\glstrifwasfirstuse\@firstoftwo
2188 \let\glscapscase\@secondofthree
2189 \glstrchecknohyperfirst{#2}%
2190 ]%
2191 {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
2192 }

```

Ensure that `\Glsfirst` honours the `nohyperfirst` attribute.

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

```

2193 \def\@GLSfirst@#1#2[#3]{%
2194 \glstrassignfieldfont{#2}%

```

Ensure that \GLSfirst honours the nohyperfirst attribute.

```
2195 \@gls@field@link
2196 [\let\glstrifwasfirstuse\@firstoftwo
2197 \let\glscapscase\@thirdofthree
2198 \glstrchecknohyperfirst{#2}%
2199 ]%
2200 {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2201 }
```

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

```
2202 \def\@glsplural@#1#2[#3]{%
2203 \glstrassignfieldfont{#2}%
2204 \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
2205 {\@gls@field@font{\glsaccessplural{#2}#3}}%
2206 }
```

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

```
2207 \def\@GLsplural@#1#2[#3]{%
2208 \glstrassignfieldfont{#2}%
2209 \@gls@field@link
2210 [\let\glsifplural\@firstoftwo
2211 \let\glscapscase\@secondofthree
2212 ]%
2213 {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}#3}}%
2214 }
```

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

```
2215 \def\@GLSplural@#1#2[#3]{%
2216 \glstrassignfieldfont{#2}%
2217 \@gls@field@link
2218 [\let\glsifplural\@firstoftwo
2219 \let\glscapscase\@thirdofthree
2220 ]%
2221 {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2222 }
```

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

```
2223 \def\glsfirstplural@#1#2[#3]{%
2224 \glstrassignfieldfont{#2}%
2225 \@gls@field@link
2226 [\let\glstrifwasfirstuse\@firstoftwo
2227 \let\glsifplural\@firstoftwo
2228 \glstrchecknohyperfirst{#2}%
2229 ]%
2230 {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
2231 }
```

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

```
2232 \def\@Glsfirstplural@#1#2[#3]{%
2233   \glstrassignfieldfont{#2}%
    Ensure that \glfirstplural honours the nohyperfirst attribute.
2234   \@gls@field@link
2235   [\let\glstrifwasfirstuse\@firstoftwo
2236   \let\glstifplural\@firstoftwo
2237   \let\glscapscase\@secondofthree
2238   \glstrchecknohyperfirst{#2}%
2239   ]%
2240   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
2241 }
```

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

```
2242 \def\@GLSfirstplural@#1#2[#3]{%
2243   \glstrassignfieldfont{#2}%
    Ensure that \glfirstplural honours the nohyperfirst attribute.
2244   \@gls@field@link
2245   [\let\glstrifwasfirstuse\@firstoftwo
2246   \let\glstifplural\@firstoftwo
2247   \let\glscapscase\@thirdofthree
2248   \glstrchecknohyperfirst{#2}%
2249   ]%
2250   {#1}{#2}%
2251   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2252 }
```

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

```
2253 \def\@glsname@#1#2[#3]{%
2254   \glstrassignfieldfont{#2}%
2255   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}#3}}%
2256 }
```

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

```
2257 \def\@Glsname@#1#2[#3]{%
2258   \glstrassignfieldfont{#2}%
2259   \@gls@field@link
2260   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2261   {\@gls@field@font{\Glsaccessname{#2}#3}}%
2262 }
```

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

```
2263 \def\@GLSname@#1#2[#3]{%
2264   \glstrassignfieldfont{#2}%
2265   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2266   {#1}{#2}%
2267   {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2268 }
```



```

\@Glsdesc@
2269 \def\@Glsdesc@#1#2[#3]{%
2270   \glstrassignfieldfont{#2}%
2271   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessdesc{#2}#3}}%
2272 }

\@GLSdesc@   First letter uppercase version.
2273 \def\@GLSdesc@#1#2[#3]{%
2274   \glstrassignfieldfont{#2}%
2275   \@gls@field@link
2276   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2277   {\@gls@field@font{\@GLSaccessdesc{#2}#3}}%
2278 }

\@GLSdesc@   All uppercase version.
2279 \def\@GLSdesc@#1#2[#3]{%
2280   \glstrassignfieldfont{#2}%
2281   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2282   {#1}{#2}{\@gls@field@font{\@GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2283 }

@Glsdescplural@   No case-changing version.
2284 \def\@Glsdescplural@#1#2[#3]{%
2285   \glstrassignfieldfont{#2}%
2286   \@gls@field@link
2287   [\let\glscapscase\@secondoftwo
2288    \let\glsifplural\@firstoftwo
2289    ]{#1}{#2}{\@gls@field@font{\@glsaccessdescplural{#2}#3}}%
2290 }

@Glsdescplural@   First letter uppercase version.
2291 \def\@Glsdescplural@#1#2[#3]{%
2292   \glstrassignfieldfont{#2}%
2293   \@gls@field@link
2294   [\let\glscapscase\@secondoftwo
2295    \let\glsifplural\@firstoftwo
2296    ]{#1}{#2}{\@gls@field@font{\@Glsaccessdescplural{#2}#3}}%
2297 }

@GLSdescplural@   All uppercase version.
2298 \def\@GLSdesc@#1#2[#3]{%
2299   \glstrassignfieldfont{#2}%
2300   \@gls@field@link
2301   [\let\glscapscase\@thirdoftwo
2302    \let\glsifplural\@firstoftwo
2303    ]%
2304    {#1}{#2}%
2305    {\@gls@field@font{\@GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
2306 }

```

\@glssymbol@

```
2307 \def\@glssymbol@#1#2[#3]{%
2308   \glstrassignfieldfont{#2}%
2309   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
2310 }
```

\@Glsymbol@ First letter uppercase version.

```
2311 \def\@Glsymbol@#1#2[#3]{%
2312   \glstrassignfieldfont{#2}%
2313   \@gls@field@link
2314   [\let\glscapscase\@secondoftwo]%
2315   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
2316 }
```

\@GLSsymbol@ All uppercase version.

```
2317 \def\@GLSsymbol@#1#2[#3]{%
2318   \glstrassignfieldfont{#2}%
2319   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2320   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
2321 }
```

lssymbolplural@ No case-changing version.

```
2322 \def\@lssymbolplural@#1#2[#3]{%
2323   \glstrassignfieldfont{#2}%
2324   \@gls@field@link
2325   [\let\glscapscase\@secondoftwo
2326   \let\glsifplural\@firstoftwo
2327   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2328 }
```

lssymbolplural@ First letter uppercase version.

```
2329 \def\@lssymbolplural@#1#2[#3]{%
2330   \glstrassignfieldfont{#2}%
2331   \@gls@field@link
2332   [\let\glscapscase\@secondoftwo
2333   \let\glsifplural\@firstoftwo
2334   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
2335 }
```

LSsymbolplural@ All uppercase version.

```
2336 \def\@LSsymbol@#1#2[#3]{%
2337   \glstrassignfieldfont{#2}%
2338   \@gls@field@link
2339   [\let\glscapscase\@thirdoftwo
2340   \let\glsifplural\@firstoftwo
2341   ]%
2342   {#1}{#2}%
2343   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2344 }
```

\@Glsuseri@ First letter uppercase version.

```
2345 \def\@Glsuseri@#1#2[#3]{%
2346   \glstrassignfieldfont{#2}%
2347   \@gls@field@link
2348   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2349   {\@gls@field@font{\Glsentryuseri{#2}#3}}%
2350 }
```

\@GLSuseri@ All uppercase version.

```
2351 \def\@GLSuseri@#1#2[#3]{%
2352   \glstrassignfieldfont{#2}%
2353   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2354   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseri{#2}#3}}}%
2355 }
```

\@Glsuserii@ First letter uppercase version.

```
2356 \def\@Glsuserii@#1#2[#3]{%
2357   \glstrassignfieldfont{#2}%
2358   \@gls@field@link
2359   [\let\glscapscase\@secondoftwo]%
2360   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
2361 }
```

\@GLSuserii@ All uppercase version.

```
2362 \def\@GLSuserii@#1#2[#3]{%
2363   \glstrassignfieldfont{#2}%
2364   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2365   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2366 }
```

\@Glsuseriii@ First letter uppercase version.

```
2367 \def\@Glsuseriii@#1#2[#3]{%
2368   \glstrassignfieldfont{#2}%
2369   \@gls@field@link
2370   [\let\glscapscase\@secondoftwo]%
2371   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
2372 }
```

\@GLSuseriii@ All uppercase version.

```
2373 \def\@GLSuseriii@#1#2[#3]{%
2374   \glstrassignfieldfont{#2}%
2375   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2376   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2377 }
```

\@Glsuseriv@ First letter uppercase version.

```
2378 \def\@Glsuseriv@#1#2[#3]{%
2379   \glstrassignfieldfont{#2}%
2380 }
```

```

2380 \@gls@field@link
2381 [\let\glscapscase\@secondoftwo]%
2382 {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
2383 }

```

\@GLSuseriv@ All uppercase version.

```

2384 \def\@GLSuseriv@#1#2[#3]{%
2385 \glstrassignfieldfont{#2}%
2386 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2387 {#1}{#2}%
2388 {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2389 }

```

\@Glsuserv@ First letter uppercase version.

```

2390 \def\@Glsuserv@#1#2[#3]{%
2391 \glstrassignfieldfont{#2}%
2392 \@gls@field@link
2393 [\let\glscapscase\@secondoftwo]%
2394 {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
2395 }

```

\@GLSuserv@ All uppercase version.

```

2396 \def\@GLSuserv@#1#2[#3]{%
2397 \glstrassignfieldfont{#2}%
2398 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2399 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2400 }

```

\@Glsuservi@ First letter uppercase version.

```

2401 \def\@Glsuservi@#1#2[#3]{%
2402 \glstrassignfieldfont{#2}%
2403 \@gls@field@link
2404 [\let\glscapscase\@secondoftwo]%
2405 {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
2406 }

```

\@GLSuservi@ All uppercase version.

```

2407 \def\@GLSuservi@#1#2[#3]{%
2408 \glstrassignfieldfont{#2}%
2409 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2410 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
2411 }

```

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

\@acrshort No case change.

```

2412 \def\@acrshort#1#2[#3]{%

```

```

2413 \glsdoifexists{#2}%
2414 {%
2415   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2416   \let\glxtrifwasfirstuse\@secondoftwo
2417   \let\gl@ifplural\@secondoftwo
2418   \let\glscapscase\@firstofthree
2419   \let\glinsert\@empty
2420   \def\glscustomtext{%
2421     \acronymfont{\glaccessshort{#2}}#3%
2422   }%
2423   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2424 }%
2425 \glspostlinkhook
2426 }

```

\@Acrshort First letter uppercase.

```

2427 \def\@Acrshort#1#2[#3]{%
2428   \glsdoifexists{#2}%
2429   {%
2430     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2431     \let\glxtrifwasfirstuse\@secondoftwo
2432     \let\gl@ifplural\@secondoftwo
2433     \let\glscapscase\@secondofthree
2434     \let\glinsert\@empty
2435     \def\glscustomtext{%
2436       \acronymfont{\Glsaccessshort{#2}}#3%
2437     }%
2438     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2439   }%
2440   \glspostlinkhook
2441 }

```

\@ACRshort All uppercase.

```

2442 \def\@ACRshort#1#2[#3]{%
2443   \glsdoifexists{#2}%
2444   {%
2445     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2446     \let\glxtrifwasfirstuse\@secondoftwo
2447     \let\gl@ifplural\@secondoftwo
2448     \let\glscapscase\@thirdofthree
2449     \let\glinsert\@empty
2450     \def\glscustomtext{%
2451       \mfirstucMakeUppercase{\acronymfont{\glaccessshort{#2}}#3}%
2452     }%
2453     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2454   }%
2455   \glspostlinkhook
2456 }

```

\@acrshortpl No case change.

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

\@Acrshortpl First letter uppercase.

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

\@ACRshortpl All uppercase.

```
2487 \def\@ACRshortpl#1#2[#3]{%
2488   \glsdoifexists{#2}%
2489   {%
2490     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2491     \let\glxtrifwasfirstuse\@secondoftwo
2492     \let\gl@sifplural\@firstoftwo
2493     \let\glscapscase\@thirdofthree
2494     \let\gl$insert\@empty
2495     \def\glscustomtext{%
2496       \mfirstucMakeUppercase{\acronymfont{\gl@saccesssshortpl{#2}}#3}%
2497     }%
2498     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2499   }%
2500   \glspostlinkhook
```

2501 }

\@acrlong No case change.

```
2502 \def\@acrlong#1#2[#3]{%
2503   \glsdoifexists{#2}%
2504   {%
2505     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2506     \let\glxtrifwasfirstuse\@secondoftwo
2507     \let\gl@sifplural\@secondoftwo
2508     \let\glscapscase\@firstofthree
2509     \let\gl$insert\@empty
2510     \def\glscustomtext{%
2511       \acronymfont{\gl@saccesslong{#2}}#3%
2512     }%
2513     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2514   }%
2515   \glspostlinkhook
2516 }
```

\@Acrlong First letter uppercase.

```
2517 \def\@Acrlong#1#2[#3]{%
2518   \glsdoifexists{#2}%
2519   {%
2520     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2521     \let\glxtrifwasfirstuse\@secondoftwo
2522     \let\gl@sifplural\@secondoftwo
2523     \let\glscapscase\@secondofthree
2524     \let\gl$insert\@empty
2525     \def\glscustomtext{%
2526       \acronymfont{\Glsaccesslong{#2}}#3%
2527     }%
2528     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2529   }%
2530   \glspostlinkhook
2531 }
```

\@ACRlong All uppercase.

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

```

2544 }%
2545 \glspostlinkhook
2546 }

```

\@acrlongpl No case change.

```

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

```

\@Acrlongpl First letter uppercase.

```

2562 \def\@Acrlongpl#1#2[#3]{%
2563   \glsdoidexists{#2}%
2564   {%
2565     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2566     \let\glxtrifwasfirstuse\@secondoftwo
2567     \let\gl@sifplural\@firstoftwo
2568     \let\glscapscase\@secondofthree
2569     \let\gl$insert\@empty
2570     \def\glscustomtext{%
2571       \acronymfont{\Glsaccesslongpl{#2}}#3%
2572     }%
2573     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2574   }%
2575   \glspostlinkhook
2576 }

```

\@ACRlongpl All uppercase.

```

2577 \def\@ACRlongpl#1#2[#3]{%
2578   \glsdoidexists{#2}%
2579   {%
2580     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2581     \let\glxtrifwasfirstuse\@secondoftwo
2582     \let\gl@sifplural\@firstoftwo
2583     \let\glscapscase\@thirdofthree
2584     \let\gl$insert\@empty
2585     \def\glscustomtext{%
2586       \mfirstucMakeUppercase{\acronymfont{\gl@saccesslongpl{#2}}#3}%

```



```

2587 }%
2588 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2589 }%
2590 \glspostlinkhook
2591 }

```

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

\@glsaddkey

```

2592 \renewcommand*{\@glsaddkey}[7]{%
2593   \key@ifundefined{glossentry}{#1}%
2594   {%
2595     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2596     \appto\@gls@keymap{,{#1}{#1}}%
2597     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
2598     \appto\@newglossaryentryposthook{%
2599       \letcs{\@glo@tmp}{@glo@#1}%
2600       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2601     }%
2602     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2603     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2604   \ifcsdef{@gls@user@#1@}%
2605   {%
2606     \PackageError{glossaries}%
2607     {Can't define '\string#5' as helper command
2608     '\expandafter\string\csname @gls@user@#1@\endcsname' already
2609     exists}%
2610     {}%
2611   }%
2612   {%
2613     \expandafter\newcommand\expandafter*\expandafter
2614     {\csname @gls@user@#1\endcsname}[2][ ]{%
2615       \new@ifnextchar[%
2616         {\csuse{@gls@user@#1@}{##1}{##2}}%
2617         {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
2618     \csdef{@gls@user@#1@}##1##2[##3]{%
2619       \@gls@field@link{##1}{##2}{#3{##2}##3}%
2620     }%
2621     \newrobustcmd*{#5}{%
2622       \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2623   }%

```

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

```

2624   \ifcsdef{@Gls@user@#1@}%
2625   {%
2626     \PackageError{glossaries}%
2627     {Can't define '\string#6' as helper command

```

```

2628         '\expandafter\string\csname @Gls@user@#1@\endcsname' already
2629         exists}%
2630     {}%
2631 }%
2632 {%
2633     \expandafter\newcommand\expandafter*\expandafter
2634     {\csname @Gls@user@#1@\endcsname}[2][\%
2635         \new@ifnextchar[%
2636             {\csuse{@Gls@user@#1@}{##1}{##2}}}%
2637             {\csuse{@Gls@user@#1@}{##1}{##2}[]}}}%
2638     \csdef{@Gls@user@#1@}##1##2[##3]{%
2639         \@gls@field@link[\let\glscapscase\@secondofthree]%
2640         {##1}{##2}{#4{##2}##3}%
2641     }%
2642     \newrobustcmd*{#6}{%
2643         \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
2644     }%

```

Finally the all caps version (modified):

```

2645     \ifcsdef{@GLS@user@#1@}%
2646     {%
2647         \PackageError{glossaries}%
2648         {Can't define '\string#7' as helper command
2649         '\expandafter\string\csname @GLS@user@#1@\endcsname' already
2650         exists}%
2651     {}%
2652 }%
2653 {%
2654     \expandafter\newcommand\expandafter*\expandafter
2655     {\csname @GLS@user@#1@\endcsname}[2][\%
2656         \new@ifnextchar[%
2657             {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2658             {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
2659     \csdef{@GLS@user@#1@}##1##2[##3]{%
2660         \@gls@field@link[\let\glscapscase\@thirdofthree]%
2661         {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2662     }%
2663     \newrobustcmd*{#7}{%
2664         \expandafter\@gls@hyp@opt\csname @GLS@user@#1@\endcsname}%
2665     }%
2666 }%
2667 {%
2668     \PackageError{glossaries-extra}{Key '#1' already exists}{}%
2669 }%
2670 }

```

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

```

2671 \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.

```
2672 \let\@glstr@org@checkfirsthyper\@gls@link@checkfirsthyper
2673 \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.

```
2674 \ifglsused{\glslabel}%
2675 {\let\glstr@ifwasfirstuse\@secondoftwo}
2676 {\let\glstr@ifwasfirstuse\@firstoftwo}%
```

Store the category label for convenience.

```
2677 \edef\glscategorylabel{\glscategory{\glslabel}}%
2678 \ifglsused{\glslabel}%
2679 {%
2680   \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
2681   {\KV@glslink@hyperfalse}{}%
2682   }%
2683   {%
2684     \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
2685     {\KV@glslink@hyperfalse}{}%
2686     }%
2687   \glslinkcheckfirsthyperhook
2688 }
```

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.

```
2689 \ifdef\do@glstdisablehyperinlist
2690 {%
2691   \let\@glstr@do@glstdisablehyperinlist\do@glstdisablehyperinlist
2692   \renewcommand*{\do@glstdisablehyperinlist}{%
2693     \@glstr@do@glstdisablehyperinlist
2694     \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}%
2695   }
2696 }
2697 {}
```

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

```
2698 \define@boolkey{glslink}{noindex}[true]{}
2699 \KV@glslink@noindexfalse
```

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

lt@glslink@opts

```
2700 \ifdef\@gls@setdefault@glslink@opts
2701 {
2702   \renewcommand*{\@gls@setdefault@glslink@opts}{%
2703     \KV@glslink@noindexfalse
```

```

2704 \glxtrsetaliasnoindex
2705 }
2706 }
2707 {

```

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

```

2708 \newcommand*{\@glstdisablehyperinlist@glslink@opts}{%
2709 \KV@glslink@noindexfalse
2710 \glxtrsetaliasnoindex
2711 }
2712 \pretoto\do@glstdisablehyperinlist{\@glstdisablehyperinlist@glslink@opts}
2713 }

```

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.)

```

2714 \providecommand*\glxtrsetaliasnoindex{%
2715 \KV@glslink@noindextrue
2716 }

```

setaliasindex

```

2717 \newcommand*{\@glxtrsetaliasindex}{%
2718 \glxtrifhasfield{alias}{\glslabel}%
2719 {%
2720 \let\glxtrindexaliased\@glxtrindexaliased
2721 \glxtrsetaliasnoindex
2722 \let\glxtrindexaliased\@no@glxtrindexaliased
2723 }%
2724 }%
2725 }

```

xtrindexaliased

```

2726 \newcommand*\@glxtrindexaliased{%
2727 \ifKV@glslink@noindex
2728 \else
2729 \begingroup
2730 \let\@glslnumberformat\@glxtr@defaultnumberformat
2731 \edef\@glscounter{\csname glo@glstdetoklabel{\glslabel}@counter\endcsname}%
2732 \glxtr@saveentrycounter
2733 \do@wrglossary{\glxtralias{\glslabel}}%
2734 \endgroup
2735 \fi
2736 }

```

xtrindexaliased

```

2737 \newcommand*\@no@glxtrindexaliased{%
2738 \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
2739 not permitted outside definition of \string\glxtrsetaliasnoindex}%
2740 {}%
2741 }

```

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

```
2742 \let\glxtrindexaliased\@no\glxtrindexaliased
```

tDefaultGlsOpts Set the default options for \glslink etc.

```
2743 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
2744   \renewcommand*{\@glslsetdefault@glslink@opts}{%
2745     \setkeys{glslink}{#1}%
2746     \@glxtrsetaliasnoindex
2747   }%
2748 }
```

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

```
2749 \newcommand*{\glxtrifindexing}[2]{%
2750   \ifKV@glslink@noindex #2\else #1\fi
2751 }
```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```
2752 \renewcommand*{\glswriteentry}[2]{%
2753   \glxtrifindexing
2754   {%
2755     \ifglindexonlyfirst
2756       \ifglused{#1}
2757       {\glxtrdoautoindexname{#1}{dualindex}}%
2758       {#2}%
2759     \else
2760       \gl@ifattribute{#1}{indexonlyfirst}{true}%
2761       {\ifglused{#1}
2762        {\glxtrdoautoindexname{#1}{dualindex}}%
2763        {#2}}%
2764       {#2}%
2765     \fi
2766   }%
2767   {}%
2768 }
```

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

```
2769 \appto\@do@wrglossary{\@glxtr@do@wrindex
2770   \glxtrdowrglossaryhook{\@glsl@label}%
2771 }
```

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

Similarly for the “noidx” version:

s@noidxglossary

```
2772 \appto\glsl@noidxglossary{\@glxtr@do@wrindex
2773   \glxtrdowrglossaryhook{\@glsl@label}%
2774 }
```

xtr@do@@wrindex

```
2775 \newcommand*{\@glsxtr@do@@wrindex}{%
2776   \glsxtrdoautoindexname{\@gls@label}{dualindex}%
2777 }
```

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.)

```
2778 \newcommand*{\glsxtrdowrglossaryhook}[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.

```
2779 \newcommand*{\@gls@alt@hyp@opt}[1]{%
2780   \let\glslinkvar\@firstofthree
2781   \let\@gls@hyp@opt@cs#1\relax
2782   \@ifstar{\s@gls@hyp@opt}%
2783   {\@ifnextchar+%
2784     {\@firstoftwo{\p@gls@hyp@opt}}%
2785     {%
2786       \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
2787       {\@firstoftwo{\@alt@gls@hyp@opt}}%
2788       {#1}%
2789     }%
2790   }%
2791 }
```

alt@gls@hyp@opt User version

```
2792 \newcommand*{\@alt@gls@hyp@opt}[1][ ]{%
2793   \let\glslinkvar\@firstofthree
2794   \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

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

```
2795 \newcommand*{\@gls@alt@hyp@opt@char}{{}
```

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

```
2796 \newcommand*{\@gls@alt@hyp@opt@keys}{{}
```

rSetAltModifier

```
2797 \newcommand*{\GlsXtrSetAltModifier}[2]{%
2798   \let\@gls@hyp@opt\@gls@alt@hyp@opt
2799   \def\@gls@alt@hyp@opt@char{#1}%
2800   \def\@gls@alt@hyp@opt@keys{#2}%
2801 }
```

org@dohyperlink

```
2802 \let\glsxtr@org@dohyperlink\glsdohyperlink
```

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

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

```
2803 \ifdef\glsnavhyperlink
2804 {
2805   \renewcommand*\glsnavhyperlink}[3][\@glo@type]{%
2806     \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
```

Scope:

```
2807   {%
2808     \let\glsdohyperlink\glsxtr@org@dohyperlink
2809     \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
2810   }%
2811 }%
2812 }
2813 {}
```

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

```
2814 \renewcommand*\glsdohyperlink}[2]{%
2815   \gls@hasattribute{\glslabel}{targeturl}%
2816   {%
2817     \gls@hasattribute{\glslabel}{targetname}%
2818     {%
2819       \gls@hasattribute{\glslabel}{targetcategory}%
2820       {%
2821         \hyperref{\gls@getattribute{\glslabel}{targeturl}}{%
2822           {\gls@getattribute{\glslabel}{targetcategory}}%
2823           {\gls@getattribute{\glslabel}{targetname}}%
2824           {\glsxtrprotectlinks#2}}%
2825       }%
2826     }%
2827     \hyperref{\gls@getattribute{\glslabel}{targeturl}}{%
2828       }%
2829       {\gls@getattribute{\glslabel}{targetname}}%
2830       {\glsxtrprotectlinks#2}}%
2831   }%
2832 }%
2833 {%
2834   \href{\gls@getattribute{\glslabel}{targeturl}}{%
2835     {\glsxtrprotectlinks#2}}%
2836 }%
```

```

2837 }%
2838 {%
    Check for alias.
2839 \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
2840 \ifvoid\gloaliaslabel
2841 {%
2842 \glsxtrhyperlink{#1}{\glsxtrprotectlinks#2}}%
2843 }%
2844 {%

```

Redirect link to the alias target.

```

2845 \glsxtrhyperlink
2846 {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
2847 {\glsxtrprotectlinks#2}}%
2848 }%
2849 }%
2850 }

```

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

```

2851 \ifdef\@glsshowtarget
2852 {
2853 \newcommand{\glsxtrhyperlink}[2]{%
2854 \@glsshowtarget{#1}%
2855 \hyperlink{#1}{#2}%
2856 }%
2857 }
2858 {
2859 \newcommand{\glsxtrhyperlink}[2]{\hyperlink{#1}{#2}}%
2860 }

```

**glsdisablehyper** Redefine to set `\glslabel` (to allow it to be picked up by `\glsdonohyperlink`). 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.

```

2861 \renewrobustcmd*{\gls hyperlink}[2][\glsentrytext{\@glo@label}]{%
2862 \glsdoifexists{#2}%
2863 {%
2864 \def\@glo@label{#2}%
2865 {\edef\glslabel{#2}%
2866 \@glslink{\glolinkprefix\glslabel}{#1}}%
2867 }%
2868 }

```

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

```

2869 \renewcommand{\glsdisablehyper}{%
2870 \KV@glslink@hyperfalse
2871 \def\@glslink{\glsdonohyperlink}%

```



```

2872 \let\@glstarget\@secondoftwo
2873 }

```

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

```

2874 \renewcommand{\glsenablehyper}{%
2875 \KV@glslink@hypertrue
2876 \def\@glslink{\glsdohyperlink}%
2877 \def\@glstarget{\glsdohypertarget}%
2878 }

```

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

```

2879 \def\glsdonohyperlink#1#2{{\glstrprotectlinks #2}}

```

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

```

2880 \ifcsundef{hyperlink}%
2881 {%
2882 \def\@glslink{\glsdonohyperlink}
2883 }%
2884 {%
2885 \def\@glslink{\glsdohyperlink}
2886 }

```

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

```

2887 \newcommand*{\glstrprotectlinks}{%
2888 \KV@glslink@hyperfalse
2889 \KV@glslink@noindextrue
2890 \let\@gls@\@glstr@p@text@
2891 \let\@Gls@\@Glsxtr@p@text@
2892 \let\@GLS@\@GLSxtr@p@text@
2893 \let\@glspl@\@glstr@p@plural@
2894 \let\@Glspl@\@Glsxtr@p@plural@
2895 \let\@GLSpl@\@GLSxtr@p@plural@
2896 \let\@glstrshort@\@glstr@p@short@
2897 \let\@Glsxtrshort@\@Glsxtr@p@short@
2898 \let\@GLSxtrshort@\@GLSxtr@p@short@
2899 \let\@glstrlong@\@glstr@p@long@
2900 \let\@Glsxtrlong@\@Glsxtr@p@long@
2901 \let\@GLSxtrlong@\@GLSxtr@p@long@
2902 \let\@glstrshortpl@\@glstr@p@shortpl@
2903 \let\@Glsxtrshortpl@\@Glsxtr@p@shortpl@
2904 \let\@GLSxtrshortpl@\@GLSxtr@p@shortpl@
2905 \let\@glstrlongpl@\@glstr@p@longpl@
2906 \let\@Glsxtrlongpl@\@Glsxtr@p@longpl@

```

```

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

```

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

@glxtrp@text@

```

2921 \def\@glxtrp@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}

```

@Glsxtrp@text@

```

2922 \def\@Glsxtrp@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}

```

@GLSxtrp@text@

```

2923 \def\@GLSxtrp@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}

```

lsxtrp@plural@

```

2924 \def\@glxtrp@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}

```

lsxtrp@plural@

```

2925 \def\@Glsxtrp@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}

```

LSxtrp@plural@

```

2926 \def\@GLSxtrp@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}

```

glxtrp@short@

```

2927 \def\@glxtrp@short@#1#2[#3]{%
2928 {%
2929   \glssetabbrvfmt{\glscategory{#2}}%
2930   \glsabbrvfont{\glsentryshort{#2}}#3%
2931 }%
2932 }

```

Glsxtrp@short@

```

2933 \def\@Glsxtrp@short@#1#2[#3]{%
2934 {%
2935   \glssetabbrvfmt{\glscategory{#2}}%
2936   \glsabbrvfont{\Glsentryshort{#2}}#3%
2937 }%
2938 }

```

GLSxtr@p@short@

```
2939 \def\@GLSxtr@p@short@#1#2[#3]{%
2940   {%
2941     \glsetabbrvfmt{\glscategory{#2}}%
2942     \mfirstucMakeUppercase{\glabbrvfont{\glentryshort{#2}}#3}%
2943   }%
2944 }
```

sxtr@p@shortpl@

```
2945 \def\@glxtr@p@shortpl@#1#2[#3]{%
2946   {%
2947     \glsetabbrvfmt{\glscategory{#2}}%
2948     \glabbrvfont{\glentryshortpl{#2}}#3%
2949   }%
2950 }
```

Sxtr@p@shortpl@

```
2951 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
2952   {%
2953     \glsetabbrvfmt{\glscategory{#2}}%
2954     \glabbrvfont{\Glsentryshortpl{#2}}#3%
2955   }%
2956 }
```

Sxtr@p@shortpl@

```
2957 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
2958   {%
2959     \glsetabbrvfmt{\glscategory{#2}}%
2960     \mfirstucMakeUppercase{\glabbrvfont{\glentryshortpl{#2}}#3}%
2961   }%
2962 }
```

@glxtr@p@long@

```
2963 \def\@glxtr@p@long@#1#2[#3]{\{\glentrylong{#2}#3}}
```

@Glsxtr@p@long@

```
2964 \def\@Glsxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}#3}}
```

@GLSxtr@p@long@

```
2965 \def\@GLSxtr@p@long@#1#2[#3]{%
2966   {\mfirstucMakeUppercase{\glslongfont{\glentrylong{#2}}#3}}}
```

lsxtr@p@longpl@

```
2967 \def\@glxtr@p@longpl@#1#2[#3]{\{\glentrylongpl{#2}#3}}
```

lsxtr@p@longpl@

```
2968 \def\@Glsxtr@p@longpl@#1#2[#3]{\{\glslongfont{\Glsentrylongpl{#2}}#3}}
```

```

LSxtr@p@longpl@
2969 \def\@GLSxtr@p@longpl@#1#2[#3]{%
2970   {\mfirstucMakeUppercase{\glslongfont{\glsentrylongpl{#2}}#3}}}

xtr@p@acrshort@
2971 \def\@glsxtr@p@acrshort@#1#2[#3]{\{\acronymfont{\glsentryshort{#2}}#3}}

xtr@p@acrshort@
2972 \def\@GLSxtr@p@acrshort@#1#2[#3]{\{\acronymfont{\Glsentryshort{#2}}#3}}

xtr@p@acrshort@
2973 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
2974   {\mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}}}

r@p@acrshortpl@
2975 \def\@glsxtr@p@acrshortpl@#1#2[#3]{\{\acronymfont{\glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
2976 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{\{\acronymfont{\Glsentryshortpl{#2}}#3}}

r@p@acrshortpl@
2977 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
2978   {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}

sxtr@p@acrlong@
2979 \def\@glsxtr@p@acrlong@#1#2[#3]{\{\glsentrylong{#2}}#3}}

sxtr@p@acrlong@
2980 \def\@GLSxtr@p@acrlong@#1#2[#3]{\{\Glsentrylong{#2}}#3}}

Sxtr@p@acrlong@
2981 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
2982   {\mfirstucMakeUppercase{\glsentrylong{#2}}#3}}}

tr@p@acrlongpl@
2983 \def\@glsxtr@p@acrlongpl@#1#2[#3]{\{\glsentrylongpl{#2}}#3}}

tr@p@acrlongpl@
2984 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{\{\Glsentrylongpl{#2}}#3}}

tr@p@acrlongpl@
2985 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
2986   {\mfirstucMakeUppercase{\glsentrylongpl{#2}}#3}}}

Commands to minimise conflict.

\@glsxtrp@opt
2987 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}

```

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

```
2988 \newcommand*{\glsxtrsetpopts}[1]{%
2989   \renewcommand*{\@glsxtrp@opt}{#1}%
2990 }
```

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

```
2991 \newcommand*{\glossxtrsetpopts}{%
2992   \glsxtrsetpopts{noindex}%
2993 }
```

`\@@glsxtrp`

```
2994 \newrobustcmd*{\@@glsxtrp}[2]{%
```

Add scope.

```
2995   {%
2996     \let\glspostlinkhook\relax
2997     \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
2998   }%
2999 }
```

`\@glsxtrp`

```
3000 \newrobustcmd*{\@glsxtrp}[2]{%
3001   \ifcsdef{gls#1}%
3002   {%
3003     \@glsxtrp{gls#1}{#2}%
3004   }%
3005   {%
3006     \ifcsdef{glsxtr#1}%
3007     {%
3008       \@glsxtrp{glsxtr#1}{#2}%
3009     }%
3010     {%
3011       \PackageError{glossaries-extra}{‘#1’ not recognised by
3012         \string\glsxtrp}{}%
3013     }%
3014   }%
3015 }
```

`\@Glsxtrp`

```
3016 \newrobustcmd*{\@Glsxtrp}[2]{%
3017   \ifcsdef{Gls#1}%
3018   {%
3019     \@glsxtrp{Gls#1}{#2}%
3020   }%
3021   {%
3022     \ifcsdef{Glsxtr#1}%
3023     {%
3024       \@glsxtrp{Glsxtr#1}{#2}%
3025     }%
3026   }%
3027 }
```

```

3026   {%
3027       \PackageError{glossaries-extra}{‘#1’ not recognised by
3028           \string\Glsxtrp}{}%
3029   }%
3030 }%
3031 }

```

\@GLSxtrp

```

3032 \newrobustcmd*{\@GLSxtrp}[2]{%
3033   \ifcsdef{GLS#1}%
3034   {%
3035       \@glsxtrp{GLS#1}{#2}%
3036   }%
3037   {%
3038       \ifcsdef{GLSxtr#1}%
3039       {%
3040           \@glsxtrp{GLSxtr#1}{#2}%
3041       }%
3042       {%
3043           \PackageError{glossaries-extra}{‘#1’ not recognised by
3044               \string\GLSxtrp}{}%
3045       }%
3046   }%
3047 }

```

\glsxtr@entry@p

```

3048 \newrobustcmd*{\glsxtr@headentry@p}[2]{%
3049   \glsifattribute{#1}{headuc}{true}%
3050   {%
3051       \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
3052   }%
3053   {%
3054       \@gls@entry@field{#1}{#2}%
3055   }%
3056 }

```

\glsxtrp Not robust as it needs to expand somewhat.

```

3057 \ifdef\texorpdfstring
3058 {
3059   \newcommand{\glsxtrp}[2]{%
3060     \protect\NoCaseChange
3061     {%
3062         \protect\texorpdfstring
3063         {%
3064             \protect\glsxtrifinmark
3065             {%
3066                 \ifcsdef{glsxtrhead#1}%
3067                 {%
3068                     {\protect\csuse{glsxtrhead#1}{#2}}%

```

```

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

```

Provide short synonyms for the most common option.

`\glsps`

```

3104 \newcommand*{\glsps}{\glstrp{short}}

```

`\glspt`

```

3105 \newcommand*{\glspt}{\glstrp{text}}

```

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

```

3106 \ifdef\teorpdfstring
3107 {
3108 \newcommand{\Glsxtrp}[2]{%

```

```

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

```

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

```

3153 \ifdef\texorpdfstring
3154 {
3155 \newcommand{\GlsXtrp}[2]{%

```



```

3156 \protect\NoCaseChange
3157 {%
3158 \protect\texorpdfstring
3159 {%
3160 \protect\glsxtrifinmark
3161 {%
3162 \ifcsdef{GLSxtr#1}%
3163 {%
3164 {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3165 }%
3166 {%
3167 \protect\mfirstucMakeUppercase
3168 {%
3169 \protect\@gls@entry@field{#2}{#1}%
3170 }%
3171 }%
3172 }%
3173 {%
3174 \@GLSxtrp{#1}{#2}%
3175 }%
3176 }%
3177 {%
3178 \protect\@gls@entry@field{#2}{#1}%
3179 }%
3180 }%
3181 }
3182 }
3183 {
3184 \newcommand{\GLSxtrp}[2]{%
3185 \protect\NoCaseChange
3186 {%
3187 \protect\glsxtrifinmark
3188 {%
3189 \ifcsdef{GLSxtr#1}%
3190 {%
3191 {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3192 }%
3193 {%
3194 \protect\mfirstucMakeUppercase
3195 {%
3196 \protect\@gls@entry@field{#2}{#1}%
3197 }%
3198 }%
3199 }%
3200 {%
3201 \@GLSxtrp{#1}{#2}%
3202 }%
3203 }%
3204 }

```

3205 }

### 1.3.5 Entry Counting

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

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

\@glsxtr@unset    Global unset.

```
3206 \newcommand*{\@glsxtr@unset}[1]{%
3207   \@glsunset{#1}%
3208   \glsxtrpostunset{#1}%
3209 }
```

\@glsunset    Global unset.

```
3210 \let\@glsunset\@glsxtr@unset
```

glsxtrpostunset

```
3211 \newcommand*{\glsxtrpostunset}[1]{}
```

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

tUnsetBuffering

```
3212 \newcommand*{\GlsXtrStartUnsetBuffering}{%
3213   \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3214 }
```

tUnsetBuffering    Unstarred version doesn't check for duplicates.

```
3215 \newcommand*{\@GlsXtrStartUnsetBuffering}{%
3216   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3217   \def\@glsxtr@unset@buffer{}%
3218   \let\@glsunset\@glsxtrbuffer@unset
3219 }
```

tUnsetBuffering    Starred version checks for duplicates.

```
3220 \newcommand*{\s@GlsXtrStartUnsetBuffering}{%
3221   \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3222   \def\@glsxtr@unset@buffer{}%
```

```

3223 \let\@glsunset\@glsxtrbuffer@nodup@unset
3224 }

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

3225 \newcommand*{\@glsxtrbuffer@unset}[1]{%
3226 \listxadd\@glsxtr@unset@buffer{#1}%
3227 }

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

3228 \newcommand*{\@glsxtrbuffer@nodup@unset}[1]{%
3229 \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}%
3230 {\listxadd\@glsxtr@unset@buffer{#1}}%
3231 }

pUnsetBuffering

3232 \newcommand*{\GlsXtrStopUnsetBuffering}{%
3233 \@ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3234 }

pUnsetBuffering Unstarred form (global unset).

3235 \newcommand*{\@GlsXtrStopUnsetBuffering}{%
3236 \let\@glsunset\@glsxtr@unset
3237 \forlistloop\@glsunset\@glsxtr@unset@buffer
3238 \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
3239 }

pUnsetBuffering Starred form (local unset).

3240 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3241 \forlistloop\@glslocalunset\@glsxtr@unset@buffer
3242 \let\@glsunset\@glsxtr@unset
3243 }

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

3244 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3245 \forlistloop#1\@glsxtr@unset@buffer
3246 }

\@glslocalunset Local unset.

3247 \renewcommand*{\@glslocalunset}[1]{%
3248 \@glslocalunset{#1}%
3249 \glsxtrpostlocalunset{#1}%
3250 }%

rpostlocalunset

3251 \newcommand*{\glsxtrpostlocalunset}[1]{%

```

```

\@glsreset   Global reset.
3252 \renewcommand*{\@glsreset}[1]{%
3253   \@glsreset{#1}%
3254   \glsxtrpostreset{#1}%
3255 }%

glsxtrpostreset
3256 \newcommand*{\glsxtrpostreset}[1]{%

\@glslocalreset   Local reset.
3257 \renewcommand*{\@glslocalreset}[1]{%
3258   \@glslocalreset{#1}%
3259   \glsxtrpostlocalreset{#1}%
3260 }%

rpostlocalreset
3261 \newcommand*{\glsxtrpostlocalreset}[1]{%

slocalreseteach   Locally reset a list of entries.
3262 \newcommand*{\glslocalreseteach}[1]{%
3263   \gls@ifnotmeasuring
3264   {%
3265     \@for\@gls@thislabel:=#1\do{%
3266       \glsdoifexists{\@gls@thislabel}%
3267       {%
3268         \@glslocalreset{\@gls@thislabel}%
3269       }%
3270     }%
3271   }%
3272 }

slocalunseteach   Locally unset a list of entries.
3273 \newcommand*{\glslocalunseteach}[1]{%
3274   \gls@ifnotmeasuring
3275   {%
3276     \@for\@gls@thislabel:=#1\do{%
3277       \glsdoifexists{\@gls@thislabel}%
3278       {%
3279         \@glslocalunset{\@gls@thislabel}%
3280       }%
3281     }%
3282   }%
3283 }

leEntryCounting   The first argument is the list of categories and the second argument is the value of the en-
trycount attribute.
3284 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%

```

Enable entry counting:

```
3285 \glsenableentrycount
```

Redefine \gls etc:

```
3286 \renewcommand*{\gls}{\cglsl}%
3287 \renewcommand*{\Gls}{\cGls}%
3288 \renewcommand*{\glspl}{\cglspl}%
3289 \renewcommand*{\Glspl}{\cGlspl}%
3290 \renewcommand*{\GLS}{\cGLS}%
3291 \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
3292 \@glsxtr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
3293 \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
3294 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3295   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3296     can't be used with \string\GlsXtrEnableEntryCounting}%
3297   {Use one or other but not both commands}}%
3298 }
```

ycountunsetattr

```
3299 \newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
3300   \@for\@glsxtr@cat:=#1\do
3301   {%
3302     \ifdefempty{\@glsxtr@cat}{}%
3303     {%
3304       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3305     }%
3306   }%
3307 }
```

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

nableentrycount

```
3308 \renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
3309 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
```

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

```
3310 \renewcommand*{\gls@defdocnewglossaryentry}{%
3311   \renewcommand*{\newglossaryentry}[2]{%
3312     \PackageError{glossaries}{\string\newglossaryentry\space
3313       may only be used in the preamble when entry counting has
3314       been activated}{If you use \string\glsenableentrycount\space
3315       you must place all entry definitions in the preamble not in
3316       the document environment}%
3317   }%
3318 }
```

New commands to access new fields:

```

3319 \newcommand*\glsentrycurrcount}[1]{%
3320 \ifcsundef{glo@glsdetoklabel{##1}@currcount}%
3321 {0}{\@gls@entry@field{##1}{currcount}}%
3322 }%
3323 \newcommand*\glsentryprevcount}[1]{%
3324 \ifcsundef{glo@glsdetoklabel{##1}@prevcount}%
3325 {0}{\@gls@entry@field{##1}{prevcount}}%
3326 }%

```

Adjust post unset and reset:

```

3327 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
3328 \renewcommand*\glsxtrpostunset}[1]{%
3329 \@glsxtr@entrycount@org@unset{##1}%
3330 \@gls@increment@currcount{##1}%
3331 }%
3332 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
3333 \renewcommand*\glsxtrpostlocalunset}[1]{%
3334 \@glsxtr@entrycount@org@localunset{##1}%
3335 \@gls@local@increment@currcount{##1}%
3336 }%
3337 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
3338 \renewcommand*\glsxtrpostreset}[1]{%
3339 \@glsxtr@entrycount@org@reset{##1}%
3340 \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
3341 }%
3342 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
3343 \renewcommand*\glsxtrpostlocalreset}[1]{%
3344 \@glsxtr@entrycount@org@localreset{##1}%
3345 \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
3346 }%

```

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

```

3347 \let\@cgl@s\@cgl@s
3348 \let\@cgl spl\@cgl spl

3349 \let\@cGls\@cGls
3350 \let\@cGlspl\@cGlspl
3351 \let\@cGLS\@cGLS
3352 \let\@cGLSpl\@cGLSpl

```

The rest is as the original definition.

```

3353 \AtEndDocument{\@gls@write@entrycounts}%
3354 \renewcommand*\@gls@entry@count}[2]{%
3355 \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
3356 }%
3357 \let\glsenableentrycount\relax
3358 \renewcommand*\glsenableentryunitcount){%
3359 \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space

```

```

3360      can't be used with \string\glsenableentrycount}%
3361      {Use one or other but not both commands}%
3362    }%
3363  }

```

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

```

3364 \newcommand*{\@gls@write@entrycounts}{%
3365   \immediate\write\@auxout
3366   {\string\providecommand*\string\@gls@entry@count}[2]{}%
3367   \count@=0\relax
3368   \forallglsentries{\@glsentry}{%
3369     \glshasattribute{\@glsentry}{entrycount}%
3370     {%
3371       \ifglsused{\@glsentry}%
3372       {%
3373         \immediate\write\@auxout
3374         {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
3375       }%
3376     }%
3377     \advance\count@ by \@ne
3378   }%
3379 }%
3380 }%
3381 \ifnum\count@=0
3382   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3383   \MessageBreak with \string\glsenableentrycount\space but the
3384   \MessageBreak attribute 'entrycount' hasn't
3385   \MessageBreak been assigned to any of the defined
3386   \MessageBreak entries}%
3387 \fi
3388 }

```

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

```

3389 \newcommand*{\glstrifcounttrigger}[3]{%
3390   \glshasattribute{#1}{entrycount}%
3391   {%
3392     \ifnum\glsentryprevcount{#1}>\glsgetattribute{#1}{entrycount}\relax
3393     #3%
3394   \else
3395     #2%
3396   \fi
3397 }%
3398 {#3}%
3399 }

```

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

\@@cgl's@

```

3400 \def\@@cgl's@#1#2[#3]{%
3401   \gl'sxtrifcounttrigger{#2}%
3402   {%
3403     \cgl'sformat{#2}{#3}%
3404     \gl'sunset{#2}%
3405   }%
3406   {%
3407     \@gl's@{#1}{#2}[#3]%
3408   }%
3409 }%
```

\@@cgl'spl@

```

3410 \def\@@cgl'spl@#1#2[#3]{%
3411   \gl'sxtrifcounttrigger{#2}%
3412   {%
3413     \cgl'splformat{#2}{#3}%
3414     \gl'sunset{#2}%
3415   }%
3416   {%
3417     \@gl'spl@{#1}{#2}[#3]%
3418   }%
3419 }%
```

\@@cGl's@

```

3420 \def\@@cGl's@#1#2[#3]{%
3421   \gl'sxtrifcounttrigger{#2}%
3422   {%
3423     \cGl'sformat{#2}{#3}%
3424     \gl'sunset{#2}%
3425   }%
3426   {%
3427     \@Gl's@{#1}{#2}[#3]%
3428   }%
3429 }%
```

\@@cGl'spl@

```

3430 \def\@@cGl'spl@#1#2[#3]{%
3431   \gl'sxtrifcounttrigger{#2}%
3432   {%
3433     \cGl'splformat{#2}{#3}%
3434     \gl'sunset{#2}%
3435   }%
3436   {%
3437     \@Gl'spl@{#1}{#2}[#3]%
3438   }%
3439 }%
```



\@@cGLS@

```
3440 \def\@@cGLS@#1#2[#3]{%
3441   \glxtrifcounttrigger{#2}%
3442   {%
3443     \cGLSformat{#2}{#3}%
3444     \glset{#2}%
3445   }%
3446   {%
3447     \@GLS@{#1}{#2}[#3]%
3448   }%
3449 }%
```

\@@cGLSpl@

```
3450 \def\@@cGLSpl@#1#2[#3]{%
3451   \glxtrifcounttrigger{#2}%
3452   {%
3453     \cGLSplformat{#2}{#3}%
3454     \glset{#2}%
3455   }%
3456   {%
3457     \@GLSpl@{#1}{#2}[#3]%
3458   }%
3459 }%
```

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

\@cgl's@

```
3460 \def\@cgl's@#1#2[#3]{\@gl's@{#1}{#2}[#3]}
```

\@cGL's@

```
3461 \def\@cGL's@#1#2[#3]{\@GL's@{#1}{#2}[#3]}
```

\@cgl'spl@

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

\@cGL'spl@

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

Add all upper case versions not provided by glossaries.

\cGLS

```
3464 \newrobustcmd*{\cGLS}{\@gl'shyp@opt\@cGLS}
```

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

```
3465 \newcommand*{\@cGLS}[2][ ]{%
3466   \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[ ]}%
3467 }
```

```

\@cGLS@
3468 \def\@cGLS@#1#2[#3]{\@GLS@{#1}{#2}[#3]}

\cGLSformat  Format used by \cGLS if entry only used once on previous run. The first argument is the label,
              the second argument is the insert text.
3469 \newcommand*\cGLSformat}[2]{%
3470   \expandafter\mfirstucMakeUppercase\expandafter{\cglformat{#1}{#2}}%
3471 }

\cGLSpl
3472 \newrobustcmd*\cGLSpl{\@gls@hyp@opt\@cGLSpl}

\@cGLSpl  Defined the un-starred form. Need to determine if there is a final optional argument
3473 \newcommand*\@cGLSpl}[2][{}]{%
3474   \new@ifnextchar[\@cGLSpl@{#1}{#2}}{\@cGLSpl@{#1}{#2}[{}]}%
3475 }

\@cGLSpl@
3476 \def\@cGLSpl@#1#2[#3]{\@GLSpl@{#1}{#2}[#3]}

\cGLSplformat  Format used by \cGLSpl if entry only used once on previous run. The first argument is the
               label, the second argument is the insert text.
3477 \newcommand*\cGLSplformat}[2]{%
3478   \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
3479 }

      Modify the trigger formats to check for the regular attribute.

\cglformat
3480 \renewcommand*\cglformat}[2]{%
3481   \glsifregular{#1}
3482   {\glsentryfirst{#1}}%
3483   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
3484 }

\cGlsformat
3485 \renewcommand*\cGlsformat}[2]{%
3486   \glsifregular{#1}
3487   {\Glsentryfirst{#1}}%
3488   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
3489 }

\cglsplformat
3490 \renewcommand*\cglsplformat}[2]{%
3491   \glsifregular{#1}
3492   {\glsentryfirstplural{#1}}%
3493   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
3494 }

```

\cGlsplformat

```
3495 \renewcommand*{\cGlsplformat}[2]{%
3496   \glsifregular{#1}
3497   {\Glsentryfirstplural{#1}}%
3498   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}\#2%
3499 }
```

New code similar to above for unit counting.

defunitcounters

```
3500 \newcommand*{\@@newglossaryentry@defunitcounters}{%
3501   \edef\@glo@countunit{\csuse{\glstr@categoryattr\@glo@category @unitcount}}%
3502   \ifvoid\@glo@countunit
3503   {}%
3504   {%
3505     \@glstr@ifunitcounter{\@glo@countunit}%
3506     {}%
3507     {\expandafter\@glstr@addunitcounter\expandafter{\@glo@countunit}}%
3508   }%
3509 }
```

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

```
3510 \newcommand*{\@glstr@unitcountlist}{{}}
```

@addunitcounter

```
3511 \newcommand*{\@glstr@addunitcounter}[1]{%
3512   \listadd{\@glstr@unitcountlist}{#1}%
3513   \ifcsundef{glstr@theunit@#1}
3514   {%
3515     \ifcsdef{theH#1}%
3516     {\csdef{glstr@theunit@#1}{\csuse{theH#1}}}%
3517     {\csdef{glstr@theunit@#1}{\csuse{the#1}}}%
3518   }%
3519   {}%
3520 }
```

r@ifunitcounter

```
3521 \newcommand*{\@glstr@ifunitcounter}[3]{%
3522   \xifinlist{#1}{\@glstr@unitcountlist}{#2}{#3}%
3523 }
```

urrentunitcount

```
3524 \newcommand*{\@glstr@currentunitcount}[1]{%
3525   glo@\glstetoklabel{#1}@currunit@\glstetattribute{#1}{unitcount}.%
3526   \csuse{glstr@theunit@\glstetattribute{#1}{unitcount}}%
3527 }
```

previousunitcount

```
3528 \newcommand*\@glsxtr@previousunitcount[1]{%
3529   glo@\glsdetoklabel{#1}@prevunit@\glsgetattribute{#1}{unitcount}.%
3530   \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3531 }
```

t@currunitcount

```
3532 \newcommand*\@gls@increment@currunitcount[1]{%
3533   \gls@hasattribute{#1}{unitcount}%
3534   {%
3535     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3536     \ifcsundef{\@glsxtr@csname}%
3537     {%
3538       \csgdef{\@glsxtr@csname}{1}%
3539       \listcsxadd
3540         {glo@\glsdetoklabel{#1}@unitlist}%
3541         {\glsgetattribute{#1}{unitcount}.%
3542         \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3543       }%
3544     }%
3545     {%
3546       \csxdef{\@glsxtr@csname}%
3547       {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
3548     }%
3549   }%
3550   {}%
3551 }
```

t@currunitcount

```
3552 \newcommand*\@gls@local@increment@currunitcount[1]{%
3553   \gls@hasattribute{#1}{unitcount}%
3554   {%
3555     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3556     \ifcsundef{\@glsxtr@csname}%
3557     {%
3558       \csdef{\@glsxtr@csname}{1}%
3559       \listcseadd
3560         {glo@\glsdetoklabel{#1}@unitlist}%
3561         {\glsgetattribute{#1}{unitcount}.%
3562         \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3563       }%
3564     }%
3565     {%
3566       \csedef{\@glsxtr@csname}%
3567       {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
3568     }%
3569   }%
3570   {}%
3571 }
```

r@currunitcount

```
3572 \newcommand*{\@glxstr@currunitcount}[2]{%
3573   \ifcsundef
3574   {glo@\glstetoklabel{#1}@currunit@#2}%
3575   {0}%
3576   {\csuse{glo@\glstetoklabel{#1}@currunit@#2}}%
3577 }%
```

r@prevunitcount

```
3578 \newcommand*{\@glxstr@prevunitcount}[2]{%
3579   \ifcsundef
3580   {glo@\glstetoklabel{#1}@prevunit@#2}%
3581   {0}%
3582   {\csuse{glo@\glstetoklabel{#1}@prevunit@#2}}%
3583 }%
```

entryunitcount

```
3584 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
3585   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
3586   \renewcommand*{\glstetocnewglossaryentry}{%
3587     \renewcommand*\newglossaryentry[2]{%
3588       \PackageError{glossaries}{\string\newglossaryentry\space
3589         may only be used in the preamble when entry counting has
3590         been activated}{If you use \string\glsenableentryunitcount\space
3591         you must place all entry definitions in the preamble not in
3592         the document environment}%
3593     }%
3594   }%
```

New commands to access new fields:

```
3595   \newcommand*{\glstentrycurrcount}[1]{%
3596     \@glxstr@currunitcount{##1}{\glstgetattribute{##1}{unitcount}}.%
3597     \csuse{glstetoc@theunit@\glstgetattribute{##1}{unitcount}}}%
3598   }%
3599   \newcommand*{\glstentryprevcount}[1]{%
3600     \@glxstr@prevunitcount{##1}{\glstgetattribute{##1}{unitcount}}.%
3601     \csuse{glstetoc@theunit@\glstgetattribute{##1}{unitcount}}}%
3602   }%
```

Access total count:

```
3603   \newcommand*{\glstentryprevtotalcount}[1]{%
3604     \ifcsundef{glo@\glstetoklabel{##1}@prevunittotal}%
3605     {0}%
3606     {%
3607       \number\csuse{glo@\glstetoklabel{##1}@prevunittotal}
3608     }%
3609   }%
```

Access max value:

```

3610 \newcommand*{\glstentryprevmaxcount}[1]{%
3611 \ifcsundef{glo@\glstetoklabel{##1}@prevunitmax}%
3612 {0}%
3613 {%
3614 \number\csuse{glo@\glstetoklabel{##1}@prevunitmax}
3615 }%
3616 }%

```

Adjust post unset and reset:

```

3617 \let\@glstxtr@entryunitcount@org@unset\glstxtrpostunset
3618 \renewcommand*{\glstxtrpostunset}[1]{%
3619 \@glstxtr@entryunitcount@org@unset{##1}%
3620 \@glst@increment@currunitcount{##1}%
3621 }%
3622 \let\@glstxtr@entryunitcount@org@localunset\glstxtrpostlocalunset
3623 \renewcommand*{\glstxtrpostlocalunset}[1]{%
3624 \@glstxtr@entryunitcount@org@localunset{##1}%
3625 \@glst@local@increment@currunitcount{##1}%
3626 }%
3627 \let\@glstxtr@entryunitcount@org@reset\glstxtrpostreset
3628 \renewcommand*{\glstxtrpostreset}[1]{%
3629 \glshasattribute{##1}{unitcount}%
3630 {%
3631 \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
3632 \ifcsundef{\@glstxtr@csname}%
3633 {}%
3634 {\csgdef{\@glstxtr@csname}{0}}%
3635 }%
3636 {}%
3637 }%
3638 \let\@glstxtr@entryunitcount@org@localreset\glstxtrpostlocalreset
3639 \renewcommand*{\glstxtrpostlocalreset}[1]{%
3640 \@glstxtr@entryunitcount@org@localreset{##1}%
3641 \glshasattribute{##1}{unitcount}%
3642 {%
3643 \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
3644 \ifcsundef{\@glstxtr@csname}%
3645 {}%
3646 {\csdef{\@glstxtr@csname}{0}}%
3647 }%
3648 {}%
3649 }%

```

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

```

3650 \let\@cglst@\@cglst@
3651 \let\@cglstpl@\@cglstpl@
3652 \let\@cGlst@\@cGlst@

```

```

3653 \let\@cGlspl@\@cGlspl@
3654 \let\@cGLS@\@cGLS@
3655 \let\@cGLSpl@\@cGLSpl@

```

Write information to the aux file.

```

3656 \AtEndDocument{\@gls@write@entryunitcounts}%
3657 \renewcommand*{\@gls@entry@unitcount}[3]{%
3658   \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
3659   \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
3660   {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
3661   {%
3662     \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
3663       \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
3664     }%
3665     \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
3666     {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
3667     {%
3668       \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
3669       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
3670     }%
3671   }%
3672 }%
3673 \let\glsenableentryunitcount\relax
3674 \renewcommand*{\glsenableentrycount}{%
3675   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
3676     can't be used with \string\glsenableentryunitcount}%
3677   {Use one or other but not both commands}%
3678 }%
3679 }
3680 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3681 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

3682 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3683   \immediate\write\@auxout
3684   {\string\@gls@entry@unitcount
3685     {\@glsentry}%
3686     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3687     }%
3688     {#1}}%
3689 }

```

entryunitcounts

```

3690 \newcommand*{\@gls@write@entryunitcounts}{%
3691   \immediate\write\@auxout
3692   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3693   \count@=0\relax

```

```

3694 \forallglentries{\@glentry}{%
3695   \glshasattribute{\@glentry}{unitcount}%
3696   {%
3697     \ifglused{\@glentry}%
3698     {%
3699       \forlistcsloop
3700         {\@gls@write@entryunitcounts@do}%
3701         {glo@\glsetoklabel{\@glentry}@unitlist}%
3702     }%
3703   }%
3704   \advance\count@ by \@ne
3705 }%
3706 {}%
3707 }%
3708 \ifnum\count@=0
3709   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3710     \MessageBreak with \string\glsenableentryunitcount\space but the
3711     \MessageBreak attribute ‘unitcount’ hasn’t
3712     \MessageBreak been assigned to any of the defined
3713     \MessageBreak entries}%
3714 \fi
3715 }

```

**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.

```

3716 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

3717 \glsenableentryunitcount

```

Redefine \gls etc:

```

3718 \renewcommand*{\gls}{\cgl}%
3719 \renewcommand*{\Gls}{\cGls}%
3720 \renewcommand*{\glspl}{\cglspl}%
3721 \renewcommand*{\Glspl}{\cGlspl}%
3722 \renewcommand*{\GLS}{\cGLS}%
3723 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3724 \@glxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

3725 \let\GlsXtrEnableEntryUnitCounting\@glxtr@setentryunitcountunsetattr
3726 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3727   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3728     can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
3729   {Use one or other but not both commands}}%
3730 }

```

**countunsetattr**



```

3731 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
3732   \@for\@glsxtr@cat:=#1\do
3733   {%
3734     \ifdefempty{\@glsxtr@cat}{}%
3735     {%
3736       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
3737       \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
3738     }%
3739   }%
3740 }

```

### 1.3.6 Acronym Modifications

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

`\SetGenericNewAcronym`

```

3741 \renewcommand*{\SetGenericNewAcronym}{%
3742   \let\@Gls@entryname\@Gls@acrenryname
3743   \renewcommand{\newacronym}[4][]{%
3744     \ifdefempty{\@glsacronymlists}%
3745     {%
3746       \def\@glo@type{\acronymtype}%
3747       \setkeys{glossentry}{##1}%
3748       \DeclareAcronymList{\@glo@type}%
3749     }%
3750   }%
3751   \glskeylisttok{##1}%
3752   \glslabeltok{##2}%
3753   \glsshorttok{##3}%
3754   \glslongtok{##4}%
3755   \newacronymhook
3756   \protected@edef\@do@newglossaryentry{%
3757     \noexpand\newglossaryentry{\the\glslabeltok}%
3758     {%
3759       type=\acronymtype,%
3760       name={\expandonce{\acronymentry{##2}}},%
3761       sort={\acronymssort{\the\glsshorttok}{\the\glslongtok}},%
3762       text={\the\glsshorttok},%
3763       short={\the\glsshorttok},%
3764       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3765       long={\the\glslongtok},%
3766       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3767       category=acronym,

```

```

3768         \GenericAcronymFields,%
3769         \the\glskeylisttok
3770     }%
3771 }%
3772 \do@newglossaryentry
3773 }%
3774 \renewcommand*{\acrfullfmt}[3]{%
3775     \glslink{##1}{##2}{\genacrfullformat{##2}{##3}}}%
3776 \renewcommand*{\Acrfullfmt}[3]{%
3777     \glslink{##1}{##2}{\Genacrfullformat{##2}{##3}}}%
3778 \renewcommand*{\ACRfullfmt}[3]{%
3779     \glslink{##1}{##2}{%
3780         \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
3781 \renewcommand*{\acrfullplfmt}[3]{%
3782     \glslink{##1}{##2}{\genplacrfullformat{##2}{##3}}}%
3783 \renewcommand*{\Acrfullplfmt}[3]{%
3784     \glslink{##1}{##2}{\Genplacrfullformat{##2}{##3}}}%
3785 \renewcommand*{\ACRfullplfmt}[3]{%
3786     \glslink{##1}{##2}{%
3787         \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
3788 \renewcommand*{\glstryfull}[1]{\genacrfullformat{##1}{}}}%
3789 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}}%
3790 \renewcommand*{\glstryfullpl}[1]{\genplacrfullformat{##1}{}}}%
3791 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}}%
3792 }

```

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:

```

3793 \let\@glstr@org@setacronymstyle\setacronymstyle
3794 \let\@glstr@org@newacronymstyle\newacronymstyle

```

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

```

3795 \newcommand*{\MakeAcronymsAbbreviations}{%
3796     \renewcommand*{\newacronym}[4][{}]{%
3797         \glstr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
3798     }%
3799     \renewcommand*{\firstacronymfont}[1]{\glstr@firstabbrvfont{##1}}%
3800     \renewcommand*{\acronymfont}[1]{\glstr@abbrvfont{##1}}%
3801     \renewcommand*{\setacronymstyle}[1]{%
3802         \PackageError{glossaries-extra}{\string\setacronymstyle{##1}}
3803         unavailable.
3804         Use \string\setabbreviationstyle\space instead.
3805         The original acronym interface can be restored with
3806         \string\RestoreAcronyms}{}%
3807     }%
3808     \renewcommand*{\newacronymstyle}[1]{%

```

```

3809 \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
3810 available unless you restore the original acronym interface with
3811 \string\RestoreAcronyms}%
3812 \@glxtr@org@newacronymstyle{##1}%
3813 }%
3814 }

```

Switch acronyms to abbreviations:

```
3815 \MakeAcronymsAbbreviations
```

**RestoreAcronyms** Restore acronyms to glossaries interface.

```

3816 \newcommand*{\RestoreAcronyms}{%
3817 \SetGenericNewAcronym
3818 \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
3819 \renewcommand{\acronymfont}[1]{##1}%
3820 \let\setacronymstyle\@glxtr@org@setacronymstyle
3821 \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.

```

3822 \renewcommand*\@gls@link@checkfirsthyper{%
3823 \ifglused{\glslabel}%
3824 {\let\glxtrifwasfirstuse\@secondoftwo}
3825 {\let\glxtrifwasfirstuse\@firstoftwo}%
3826 \@glxtr@org@checkfirsthyper
3827 }
3828 \glssetcategoryattribute{acronym}{regular}{false}%
3829 \setacronymstyle{long-short}%
3830 }

```

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

```

3831 \renewcommand*{\glsacspace}[1]{%
3832 \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
3833 \ifdim\dimen@<\glsacspacemax~\else\space\fi
3834 }

```

**\glsacspacemax** Value used in the above.

```
3835 \newcommand*{\glsacspacemax}{3em}
```

### 1.3.7 Indexing and Displaying Glossaries

From time-to-time users ask if they can have one glossary sorted normally and another sorted by definition or usage. With the base glossaries package this can only be achieved with the “noidx” commands (Option 1). This is an attempt to mix and match.

First we need a list of the glossaries that require `makeindex/xindy`.

**r@reg@glosslist**

```
3836 \newcommand*{\@glxtr@reg@glosslist}{}
```

Save the original definition of `\makeglossaries`:

```
3837 \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
```

```
3838 \renewcommand*{\makeglossaries}[1] [] {%
3839   \ifx\@glxtr@record@setting\@glxtr@record@setting@only
3840     \PackageError{glossaries-extra}{\string\makeglossaries\space
3841       not permitted\MessageBreak with record=only package option}%
3842     {You may only use \string\makeglossaries\space with
3843       record=off or record=alsoindex options}%
3844   \else
3845     \ifblank{#1}%
3846     {\@glxtr@org@makeglossaries}%
3847     {%
3848       \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
3849         \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
3850           not permitted\MessageBreak with record=alsoindex package option}%
3851         {You may only use the hybrid \string\makeglossaries[...]\space with
3852           record=off option}%
3853       \else
3854         \edef\@glxtr@reg@glosslist{#1}%
3855         \ifundef{\glswrite}{\newwrite\glswrite}{}%
3856         \protected@write\@auxout{}{\string\providecommand
3857           \string\@glxtr@order[1]{}%
3858           \protected@write\@auxout{}{\string\providecommand
3859             \string\@istfilename[1]{}%
3860             \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
3861             \protected@write\@auxout{}{\string\@glxtr@order{\glxtr@order}}%
3862             \protected@write\@auxout{}{\string\glxtr@makeglossaries{#1}}%
3863             \write\@auxout{\string\providecommand\string\@glxtr@reference[3]{}}}%
3864       \for\@glo@type:=#1\do{%
3865         \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
3866       }%
3867     }%
3868   \PackageError{glossaries}{New glossaries
3869     must be created before \string\makeglossaries}{You need
3870     to move \string\makeglossaries\space after all your
3871     \string\newglossary\space commands}}%
```

Iterate through each supplied glossary type and activate it.

```
3864   \for\@glo@type:=#1\do{%
3865     \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
3866   }%
```

New glossaries must be created before `\makeglossaries`:

```
3867   \renewcommand*\newglossary[4] [] {%
3868     \PackageError{glossaries}{New glossaries
3869       must be created before \string\makeglossaries}{You need
3870       to move \string\makeglossaries\space after all your
3871       \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
3872   \let\@makeglossary\relax
3873   \let\makeglossary\relax
```

```

3874      \renewcommand\makeglossaries[1][]{}%
Disable all commands that have no effect after \makeglossaries
3875      \@disable@onlypremakeg
Allow see key:
3876      \let\gls@checkseeallowed\relax
Adjust \@do@seeglossary. This needs to check for the entry's existence but don't increment
associated counter.
3877      \renewcommand*\@do@seeglossary}[2]{%
3878      \glsdoifexists{##1}%
3879      {%
3880      \edef\@gls@label{\glsdetoklabel{##1}}%
3881      \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3882      \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3883      {\@glsxtr@org@doseeglossary{##1}{##2}}%
3884      {%
3885      \@@glsxtrwrglossmark
3886      \protected@write\@auxout{}{%
3887      \string\@gls@reference
3888      {\@gls@type}{\@gls@label}{\string\glsseeformat##2}}}%
3889      }%
3890      }%
3891      }%
3892      }%
Adjust \@do@@wrglossary
3893      \let\@glsxtr@@do@@wrglossary\@do@@wrglossary
3894      \def\@do@@wrglossary{%
3895      \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3896      \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3897      {\@glsxtr@@do@@wrglossary}%
3898      {\@gls@noidxglossary}%
3899      }%
Suppress warning about no \makeglossaries
3900      \let\warn@nomakeglossaries\relax
3901      \def\warn@noprintglossary{%
3902      \GlossariesWarningNoLine{No \string\printglossary\space
3903      or \string\printglossaries\space
3904      found.^^J(Remove \string\makeglossaries\space if you don't want
3905      any glossaries.)^^JThis document will not have a glossary}%
3906      }%
Only warn for glossaries not listed.
3907      \renewcommand{\@gls@noref@warn}[1]{%
3908      \edef\@gls@type{##1}%
3909      \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3910      {%
3911      \GlossariesExtraWarning{Can't use
3912      \string\printnoidxglossary[type={\@gls@type}]

```

```

3913         when '@gls@type' is listed in the optional argument of
3914         \string\makeglossaries}%
3915     }%
3916     {%
3917         \GlossariesWarning{Empty glossary for
3918         \string\printnoidxglossary[type={##1}].
3919         Rerun may be required (or you may have forgotten to use
3920         commands like \string\gls)}%
3921     }%
3922 }%

```

Adjust display number list to check for type:

```

3923 \renewcommand*{\glsdisplaynumberlist}[1]{%
3924 \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3925 {\@glsxtr@idx@displaynumberlist{##1}}%
3926 {\@glsxtr@noidx@displaynumberlist{##1}}%
3927 }%

```

Adjust entry list:

```

3928 \renewcommand*{\glsentrynumberlist}[1]{%
3929 \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3930 {\@glsxtr@idx@entrynumberlist{##1}}%
3931 {\@glsxtr@noidx@entrynumberlist{##1}}%
3932 }%

```

Adjust number list loop

```

3933 \renewcommand*{\glsnumberlistloop}[2]{%
3934 \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3935 {%
3936     \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3937     not available for glossary '@##1'}{%
3938     }%
3939     {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
3940 }%

```

Only sanitize sort for normal indexing glossaries.

```

3941 \renewcommand*{\glsprestandardsort}[3]{%
3942 \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
3943 {%
3944     \glsdosanitizesort
3945 }%
3946 {%
3947     \ifglssanitizesort
3948     \@gls@noidx@sanitizesort
3949     \else
3950     \@gls@noidx@nosanitizesort
3951     \fi
3952 }%
3953 }%

```

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

```

3954 \renewcommand*\new@glossaryentry[2]{%
3955 \PackageError{glossaries-extra}{Glossary entries must be defined
3956 in the preamble\MessageBreak when you use the optional argument
3957 of \string\makeglossaries}{Either move your definitions to the
3958 preamble or don't use the optional argument of
3959 \string\makeglossaries}%
3960 }%

```

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).

```

3961 \let\@glo@assign@sortkey\@glxtr@mixed@assign@sortkey
3962 \renewcommand*\@printgloss@setsort{%

```

Need to extract just the type value.

```

3963 \expandafter\@glxtr@gettype\expandafter,\@glxtr@printglossopts,%
3964 type=\glsdefaulttype,\@end@glxtr@gettype
3965 \def\@glo@sorttype{\@glo@default@sorttype}%
3966 }%

```

Check automake setting:

```

3967 \ifglautomake
3968 \renewcommand*\@gls@doautomake{%
3969 \@for\@gls@type:=\@glxtr@reg@glosslist\do{%
3970 \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
3971 }%
3972 }%
3973 \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

3974 \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
3975 \fi
3976 }%
3977 \fi
3978 }

```

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.)

```

3979 \newcommand{\@glxtr@orgprintglossary}[2]{%
3980 \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

3981 \def\glossarytitle{%
3982 \ifcsdef{\@glo@type@\@glo@type @title}%
3983 {\csuse{\@glo@type@\@glo@type @title}}%
3984 {\glossaryname}}%
3985 \def\glossarytoctitle{\glossarytitle}%

```

```

3986 \let\org@glossarytitle\glossarytitle
3987 \def\@glossarystyle{%
3988   \ifx\@glossary@default@style\relax
3989     \GlossariesWarning{No default glossary style provided \MessageBreak
3990       for the glossary '\@glo@type'. \MessageBreak
3991       Using deprecated fallback. \MessageBreak
3992       To fix this set the style with \MessageBreak
3993       \string\setglossarystyle\space or use the \MessageBreak
3994       style key=value option}%
3995   \fi
3996 }%
3997 \def\gls@dotoc@title{\gls@settoc@title{\@glo@type}}%
3998 \let\org@glossaryentrynumbers\glossaryentrynumbers
3999 \bgroup
4000   \@printgloss@setsort
4001   \setkeys{printgloss}{#1}%
4002   \ifx\glossarytitle\org@glossarytitle
4003   \else
4004     \cslet{\@glo@type@\@glo@type @title}{\glossarytitle}%
4005   \fi
4006   \let\currentglossary\@glo@type
4007   \let\org@glossaryentrynumbers\glossaryentrynumbers
4008   \let\glsnonextpages\@glsnonextpages
4009   \let\glsnextpages\@glsnextpages

4010   \glsxtractivatenopost
4011   \gls@dotoc@title
4012   \@glossarystyle
4013   \let\gls@org@glossaryentryfield\glossentry
4014   \let\gls@org@glossarysubentryfield\subglossentry
4015   \renewcommand{\glossentry}[1]{%
4016     \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4017     \gls@org@glossaryentryfield{##1}%
4018   }%
4019   \renewcommand{\subglossentry}[2]{%
4020     \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4021     \gls@org@glossarysubentryfield{##1}{##2}%
4022   }%
4023   \@gls@preglossaryhook
4024   #2%
4025 \egroup
4026 \global\let\glossaryentrynumbers\org@glossaryentrynumbers
4027 \global\let\warn@noprntglossary\relax
4028 }

```

ractivatenopost Change \nopostdesc and \glsxtrnopostpunc to behave as they do in the glossary.

```

4029 \newcommand*{\glsxtractivatenopost}{%
4030   \let\nopostdesc\@nopostdesc
4031   \let\glsxtrnopostpunc\@glsxtr@nopostpunc
4032 }

```



lsxtrnopostpunc

```
4033 \newrobustcmd*{\glxtrnopostpunc}{}
```

lsxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```
4034 \newcommand{\@glxtr@nopostpunc}{%
4035   \let\@glxtr@org@postdescription\glspostdescription
4036   \ifglsnopostdot
4037     \renewcommand{\glspostdescription}{%
4038       \glsnopostdottrue
4039       \let\glspostdescription\@glxtr@org@postdescription
4040       \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4041       \glxtrpostdescription
4042       \@glxtr@nopostpunc@postdesc}%
4043   \else
4044     \renewcommand{\glspostdescription}{%
4045       \let\glspostdescription\@glxtr@org@postdescription
4046       \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4047       \glxtrpostdescription
4048       \@glxtr@nopostpunc@postdesc}%
4049   \fi
4050   \glsnopostdotfalse
4051 }
```

stpunc@postdesc

```
4052 \newcommand*{\@glxtr@nopostpunc@postdesc}{}
```

restore@postpunc

```
4053 \newcommand*{\@glxtr@restore@postpunc}{%
4054   \def\@glxtr@nopostpunc@postdesc{%
4055     \@glxtr@org@postdescription
4056     \let\@glxtr@nopostpunc@postdesc\@empty
4057     \let\glxtrrestorepostpunc\@empty
4058   }%
4059 }
```

restorepostpunc Does nothing outside of glossary.

```
4060 \newcommand*{\glxtrrestorepostpunc}{}
```

\@printglossary Redefine.

```
4061 \renewcommand{\@printglossary}[2]{%
4062   \def\@glxtr@printglossopts{#1}%
4063   \@glxtr@orgprintglossary{#1}{#2}%
4064 }
```

Add a key that switches off the entry targets:

```
4065 \define@choicekey{printgloss}{target}
4066 [\@glxtr@printglossval\@glxtr@printglossnr]%
```

```

4067 {true,false}[true]%
4068 {%
4069   \ifcase\@glxtr@printglossnr

4070   \def\@glstarget{\glsdohypertarget}%
4071   \else
4072   \let\@glstarget\@secondoftwo
4073   \fi
4074 }

```

hypernameprefix

```

4075 \newcommand{\@glxtrhypernameprefix}{}

```

New to v1.20:

```

4076 \define@key{printgloss}{targetnameprefix}{%
4077   \renewcommand{\@glxtrhypernameprefix}{#1}%
4078 }

4079 \define@key{printgloss}{prefix}{%
4080   \renewcommand{\glo@linkprefix}{#1}%
4081 }

```

glsdohypertarget Redefine to insert \@glxtrhypernameprefix before the target name.

```

4082 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
4083 \renewcommand{\glsdohypertarget}[2]{%
4084   \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
4085 }

```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```

4086 \ifx\@glstarget\@glxtr@org@glsdohypertarget
4087   \def\@glstarget{\glsdohypertarget}%
4088 \fi
4089 %\end{macro}

```

@makeglossaries For the benefit of makeglossaries

```

4090 \newcommand*{\@glxtr@makeglossaries}[1]{}

```

@glxtr@gettype Get just the type.

```

4091 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
4092   \def\@glo@type{#2}%
4093 }

```

@assign@sortkey Assign the sort key.

```

4094 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
4095   \edef\@glo@type{\@glo@type}%
4096   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
4097   {%
4098     \@glo@no@assign@sortkey{#1}%

```

```

4099 }%
4100 {%
4101   \@@glo@assign@sortkey{#1}%
4102 }%
4103 }%

```

Display number list for the regular version:

splaynumberlist

```

4104 \let\@glstr@idx@displaynumberlist\glsdisplaynumberlist

```

Display number list for the “noidx” version:

splaynumberlist

```

4105 \newcommand*{\@glstr@noidx@displaynumberlist}[1]{%
4106   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4107   \ifdef\@gls@loclist
4108     {%
4109       \def\@gls@noidxloclist@sep{%
4110         \def\@gls@noidxloclist@sep{%
4111           \def\@gls@noidxloclist@sep{%
4112             \glsnumlistsep
4113           }%
4114           \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
4115         }%
4116       }%
4117       \def\@gls@noidxloclist@finalsep{}%
4118       \def\@gls@noidxloclist@prev{}%
4119       \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4120       \@gls@noidxloclist@finalsep
4121       \@gls@noidxloclist@prev
4122     }%
4123     {%
4124       \glstrundeftag
4125       \glsdoifexists{#1}%
4126       {%
4127         \GlossariesWarning{Missing location list for ‘#1’. Either
4128           a rerun is required or you haven’t referenced the entry.}%
4129       }%
4130     }%
4131 }%
4132

```

And for the number list loop:

@numberlistloop

```

4133 \newcommand*{\@glstr@noidx@numberlistloop}[3]{%
4134   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4135   \let\@gls@org\glsnoidxdisplayloc\glsnoidxdisplayloc
4136   \let\@gls@org\glsseeformat\glsseeformat

```

```

4137 \let\glsnoidxdisplayloc#2\relax
4138 \let\glsseeformat#3\relax
4139 \ifdef\@gls@loclist
4140 {%
4141   \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4142 }%
4143 {%

4144   \glsxtrundeftag
4145   \glsdoifexists{#1}%
4146   {%
4147     \GlossariesWarning{Missing location list for ‘##1’. Either
4148       a rerun is required or you haven’t referenced the entry.}%
4149   }%
4150 }%
4151 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4152 \let\glsseeformat\@gls@org@glsseeformat
4153 }%

```

Same for entry number list.

entrynumberlist

```

4154 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4155   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4156   \ifdef\@gls@loclist
4157   {%
4158     \glsnoidxloclist{\@gls@loclist}%
4159   }%
4160   {%

4161     \glsxtrundeftag
4162     \glsdoifexists{#1}%
4163     {%
4164       \GlossariesWarning{Missing location list for ‘#1’. Either
4165         a rerun is required or you haven’t referenced the entry.}%
4166     }%
4167   }%
4168 }%

```

entrynumberlist

```

4169 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

4170 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4171   \protected@edef\@glsxtr@titlelabel{#1}%
4172   \ifdefvoid\@glsxtr@titlelabel
4173   {%
4174   {%
4175     \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4176   }%

```

```

4177 \ifdefvoid{\@glsxtr@titlelabel}%
4178 {%
4179   \DTLifint{#1}%
4180   {%
4181     \ifnum#1<256\relax
4182       \edef#2{\char#1\relax}%
4183     \else
4184       \edef#2{#1}%
4185     \fi
4186   }%
4187   {%
4188     \ifcsundef{#1groupname}%
4189     {\def#2{#1}}%
4190     {\letcs#2{#1groupname}}%
4191   }%
4192 }%
4193 {%
4194   \let#2\@glsxtr@titlelabel
4195 }%
4196 }

```

**g@getgrouptitle** Save original definition of \@gls@getgrouptitle

```

4197 \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.

```

4198 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4199   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4200   \@onelevel@sanitize\@glsxtr@titlelabel
4201   \ifcsdef{\@glsxtr@titlelabel}
4202   {\letcs{#2}{\@glsxtr@titlelabel}}%
4203   {\glsxtr@org@getgrouptitle{#1}{#2}}%
4204 }
4205 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

**trsetgrouptitle** Sets the title for the given group label.

```

4206 \newcommand{\glsxtrsetgrouptitle}[2]{%
4207   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4208   \@onelevel@sanitize\@glsxtr@titlelabel
4209   \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4210 }

```

**alsetgrouptitle** As above put only locally defines the title.

```

4211 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
4212   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4213   \@onelevel@sanitize\@glsxtr@titlelabel
4214   \protected@csedef{\@glsxtr@titlelabel}{#2}%
4215 }

```

`\glsnavigation`   Redefine to use new user-level command.

```
4216 \renewcommand*{\glsnavigation}{%
4217   \def\@gls@between{}%
4218   \ifcsundef\@gls@hypergroup\list@{\@glo@type}%
4219   {%
4220     \def\@gls@list{}%
4221   }%
4222   {%
4223     \expandafter\let\expandafter\@gls@list
4224       \csname @gls@hypergroup\list@{\@glo@type}\endcsname
4225   }%
4226   \@for\@gls@tmp:=\@gls@list\do{%
4227     \@gls@between
4228     \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4229     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4230     \let\@gls@between\glshypernavsep
4231   }%
4232 }
```

`@noidx@glossary`

```
4233 \renewcommand*{\@print@noidx@glossary}{%
4234   \ifcsdef\@glsref@{\@glo@type}%
4235   {%
4236     \ifcsdef\@glo@sortmacro@{\@glo@sorttype}%
4237     {%
4238       \csuse{\@glo@sortmacro@{\@glo@sorttype}}{\@glo@type}%
4239     }%
4240     {%
4241       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4242     }%
4243     \glossarysection[\glossarytoctitle]{\glossarytitle}%
4244     \glossarypreamble
```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```
4245   \def\@gls@currentlettergroup{}%
4246   \begin{theglossary}%
4247     \glossaryheader
4248     \glsresetentrylist
4249     \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}%
4250     \end{theglossary}%
4251     \glossarypostamble
4252   }%
4253   {%
```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```
4254     \glsxtrifemptyglossary{\@glo@type}%
4255     {}%
4256     {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
```

```

4257 \@gls@noref@warn{\@glo@type}%
4258 }%
4259 }

```

noidxdisplayloc Patch to check for range formations.

```

4260 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4261 \setentrycounter[#1]{#2}%
4262 \@glsxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
4263 }

```

xtr@display@loc Patch to check for range formations.

```

4264 \def\@glsxtr@display@loc#1#2\end@glxtr@display@loc#3{%
4265 \ifx#1\relax
4266 \glsxtrdisplaystartloc{#2}{#3}%
4267 \else
4268 \ifx#1\relax
4269 \glsxtrdisplayendloc{#2}{#3}%
4270 \else
4271 \glsxtrdisplaysingleloc{#1#2}{#3}%
4272 \fi
4273 \fi
4274 }

```

isplaysingleloc Single location.

```

4275 \newcommand*{\glsxtrdisplaysingleloc}[2]{%
4276 \csuse{#1}{#2}%
4277 }

```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glsxtrlocrangefmt`.

displaystartloc Start of a location range.

```

4278 \newcommand*{\glsxtrdisplaystartloc}[2]{%
4279 \edef\glsxtrlocrangefmt{#1}%
4280 \ifx\glsxtrlocrangefmt\empty
4281 \def\glsxtrlocrangefmt{glsnumberformat}%
4282 \fi
4283 \expandafter\glsxtrdisplaysingleloc
4284 \expandafter{\glsxtrlocrangefmt}{#2}%
4285 }

```

trdisplayendloc End of a location range.

```

4286 \newcommand*{\glsxtrdisplayendloc}[2]{%
4287 \edef\@glsxtr@tmp{#1}%
4288 \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}}%
4289 \ifx\glsxtrlocrangefmt\@glsxtr@tmp
4290 \else
4291 \GlossariesExtraWarning{Mismatched end location range
4292 (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%

```

```

4293 \fi
4294 \expandafter\glxtrdisplayendlochook\expandafter{\@glxtr@tmp}{#2}%
4295 \expandafter\glxtrdisplaysingleloc
4296 \expandafter{\glxtrlocrangefmt}{#2}%
4297 \def\glxtrlocrangefmt{%
4298 }

```

`\displayendlochook` Allow the user to hook into the end of range command.

```

4299 \newcommand*{\glxtrdisplayendlochook}[2]{%

```

`\glxtrlocrangefmt` Current range format. Empty if not in a range.

```

4300 \newcommand*{\glxtrlocrangefmt}{%

```

`\setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```

4301 \renewcommand*{\setentrycounter}[2][{}]{%
4302 \def\glxtrcounterprefix{#1}%
4303 \ifx\glxtrcounterprefix\@empty
4304 \def\@glo@counterprefix{.}%
4305 \else
4306 \def\@glo@counterprefix{.#1.}%
4307 \fi
4308 \def\glsetentrycounter{#2}%
4309 }

```

`\ls@removespaces` Redefine to allow adjustments to location hyperlink.

```

4310 \def\@gls@removespaces#1 #2\@nil{%
4311 \toks@=\expandafter{\the\toks@#1}%
4312 \ifx\@#2\%
4313 \edef\x{\the\toks@}%
4314 \ifx\x\empty
4315 \else
4316 \expandafter\glxtrlocationhyperlink\expandafter
4317 \glsetentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4318 \fi
4319 \else
4320 \@gls@ReturnAfterFi{%
4321 \@gls@removespaces#2\@nil
4322 }%
4323 \fi
4324 }

```

`\locationhyperlink`

```

4325 \newcommand*{\glxtrlocationhyperlink}[3]{%
4326 \ifdefined\glxtrsupplocationurl
4327 {%
4328 \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4329 }%

```



```

4330 {%
4331   \hyperref{\glxtrsupplocationurl}{\#1\#2\#3}{\#3}%
4332 }%
4333 }

```

supphypernumber

```

4334 \newcommand*\glxtrsupphypernumber}[1]{%
4335 {%
4336   \glshasattribute{\glscurrententrylabel}{externallocation}%
4337   {%
4338     \def\glxtrsupplocationurl{%
4339       \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4340   }%
4341   {%
4342     \def\glxtrsupplocationurl{%
4343     }%
4344     \glshypernumber{\#1}%
4345   }%
4346 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

4347 \renewcommand*\@print@glossary{%
4348   \makeatletter
4349   \@input{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4350   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4351   {}%
4352   {\glxtrNoGlossaryWarning{\@glo@type}}%
4353   \ifglxindy
4354     \ifcsundef{\xdy@\@glo@type @language}%
4355     {%
4356       \edef\@do@auxoutstuff{%
4357         \noexpand\AtEndDocument{%
4358           \noexpand\immediate\noexpand\write\@auxout{%
4359             \string\providecommand\string\@xdylanguage[2]{}}%
4360           \noexpand\immediate\noexpand\write\@auxout{%
4361             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4362         }%
4363       }%
4364     }%
4365     {%
4366       \edef\@do@auxoutstuff{%
4367         \noexpand\AtEndDocument{%
4368           \noexpand\immediate\noexpand\write\@auxout{%
4369             \string\providecommand\string\@xdylanguage[2]{}}%
4370           \noexpand\immediate\noexpand\write\@auxout{%
4371             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type

```

```

4372         @language\endcsname}}}%
4373     }%
4374 }%
4375 }%
4376 \do@auxoutstuff
4377 \edef\do@auxoutstuff{%
4378     \noexpand\AtEndDocument{%
4379         \noexpand\immediate\noexpand\write\@auxout{%
4380             \string\providecommand\string\@gls@codepage[2]{}}%
4381         \noexpand\immediate\noexpand\write\@auxout{%
4382             \string\@gls@codepage{\@glo@type}{\@gls@codepage}}}%
4383     }%
4384 }%
4385 \do@auxoutstuff
4386 \fi
4387 \renewcommand*{\@warn@nomakeglossaries}{%
4388     \GlossariesWarningNoLine{\string\makeglossaries\space
4389     hasn't been used,^^Jthe glossaries will not be updated}%
4390 }%
4391 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`oGlsWarningHead` Header message.

```

4392 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4393     This document is incomplete. The external file associated with
4394     the glossary '#1' (which should be called \texttt{#2})
4395     hasn't been created.%
4396 }

```

`arningEmptyStart` No entries have been added to the glossary.

```

4397 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4398     This has probably happened because there are no entries defined
4399     in this glossary.%
4400 }

```

`arningEmptyMain` The default “main” glossary is empty.

```

4401 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4402     If you don't want this glossary,
4403     add \texttt{nomain} to your package option list when you load
4404     \texttt{glossaries-extra.sty}. For example:%
4405 }

```

`ingEmptyNotMain` A glossary that isn't the default “main” glossary is empty.

```

4406 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4407     Did you forget to use \texttt{type=#1} when you defined your
4408     entries? If you tried to load entries into this glossary with
4409     \texttt{\string\loadglsentries} did you remember to use
4410     \texttt{[#1]} as the optional argument? If you did, check that

```

```

4411 the definitions in the file you loaded all had the type set
4412 to \texttt{\string\glsdefaulttype}.%
4413 }

```

WarningCheckFile Advisory message to check the file contents.

```

4414 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4415   Check the contents of the file \texttt{#1}. If
4416   it's empty, that means you haven't indexed any of your entries in this
4417   glossary (using commands like \texttt{\string\gls} or
4418   \texttt{\string\glsadd}) so this list can't be generated.
4419   If the file isn't empty, the document build process hasn't been
4420   completed.%
4421 }

```

WarningAutoMake Message when automake option has been used.

```

4422 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4423   You may need to rerun \LaTeX. If you already have, it may be that
4424   \TeX's shell escape doesn't allow you to run
4425   \ifglxindy xindy\else makeindex\fi. Check the
4426   transcript file \texttt{\jobname.log}. If the shell escape is
4427   disabled, try one of the following:
4428
4429   \begin{itemize}
4430     \item Run the external (Lua) application:
4431
4432       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4433
4434     \item Run the external (Perl) application:
4435
4436       \texttt{makeglossaries \string"\jobname\string"}
4437   \end{itemize}
4438
4439   Then rerun \LaTeX\ on this document.
4440   \GlossariesExtraWarning{Rerun required to build the
4441   glossary '#1' or check TeX's shell escape allows
4442   you to run \ifglxindy xindy\else makeindex\fi}%
4443 }

```

WarningMisMatch Mismatching \makenoidxglossaries.

```

4444 \newcommand{\GlsXtrNoGlsWarningMisMatch}{%
4445   You need to either replace \texttt{\string\makenoidxglossaries}
4446   with \texttt{\string\makeglossaries} or replace
4447   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4448   \texttt{\string\printnoidxglossary}
4449   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4450   this document.%
4451 }

```

arningBuildInfo Build advice.

```
4452 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4453   Try one of the following:
4454   \begin{itemize}
4455     \item Add \texttt{automake} to your package option list when you load
4456       \texttt{glossaries-extra.sty}. For example:
4457
4458       \texttt{\string\usepackage[automake]%
4459         \glsopenbrace glossaries-extra\glsclosebrace}
4460
4461     \item Run the external (Lua) application:
4462
4463     \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4464
4465     \item Run the external (Perl) application:
4466
4467     \texttt{makeglossaries \string"\jobname\string"}
4468   \end{itemize}
4469
4470   Then rerun \LaTeX\ on this document.%
4471 }
```

trRecordWarning Paragraph for record=only.

```
4472 \newcommand{\GlsXtrRecordWarning}[1]{%
4473   \texttt{\string\printglossary} doesn't work
4474   with the \texttt{record=only} package option
4475   use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4476   instead (or change the package option).%
4477 }
```

oGlsWarningTail Final paragraph.

```
4478 \newcommand{\GlsXtrNoGlsWarningTail}{%
4479   This message will be removed once the problem has been fixed.%
4480 }
```

GlsWarningNoOut No out file created. Build advice.

```
4481 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4482   The file \texttt{#1} doesn't exist. This most likely means you haven't used
4483   \texttt{\string\makeglossaries} or you have used
4484   \texttt{\string\nofiles}. If this is just a draft version of the
4485   document, you can suppress this message using the
4486   \texttt{nomissingglstext} package option.%
4487 }
```

glossarywarning

```
4488 \newcommand*{\@glsxtr@defaultnoglossarywarning}[1]{%
4489   \glossarysection[\glossarytoctitle]{\glossarytitle}
4490   \GlsXtrNoGlsWarningHead{#1}{\jobname.\cstype @glo@type @in\endcsname}
4491   \par
```

```

4492 \glstrifemptyglossary{#1}%
4493 {%
4494     \GlsXtrNoGlsWarningEmptyStart\space
4495     \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4496     \medskip
4497     \noindent\texttt{\string\usepackage[nomain\ifglssacronym ,acronym\fi]%
4498         \glsoopenbrace glossaries-extra\glsclosebrace}
4499     \medskip
4500     }%
4501     {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4502 }%
4503 {%
4504     \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4505     {%
4506         \GlsXtrNoGlsWarningCheckFile
4507         {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4508
4509         \ifglssautomake
4510
4511         \GlsXtrNoGlsWarningAutoMake{#1}
4512
4513         \else
4514
4515         \ifthenelse{\equal{#1}{main}}{%
4516             {%
4517                 \GlsXtrNoGlsWarningEmptyMain\par
4518                 \medskip
4519                 \noindent\texttt{\string\usepackage[nomain]%
4520                     \glsoopenbrace glossaries-extra\glsclosebrace}
4521                 \medskip
4522             }%
4523             {}%
4524
4525             \ifdefequal\makeglossaries\@no@makeglossaries
4526             {%
4527                 \GlsXtrNoGlsWarningMisMatch
4528             }%
4529             {%
4530                 \GlsXtrNoGlsWarningBuildInfo
4531             }%
4532         \fi
4533     }%
4534     {%
4535         \GlsXtrNoGlsWarningNoOut
4536         {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4537     }%
4538 }%
4539 \par
4540 \GlsXtrNoGlsWarningTail

```

4541 }

glossarywarning Warn about using \printglossary with record

```
4542 \newcommand*{\@glstr@record@noglossarywarning}[1]{%
4543   \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
4544   with record=only package option\MessageBreak(use
4545   \string\printunsrtglossary[type=#1])\MessageBreak
4546   instead (or change the package option)}%
4547   \glossarysection[\glossarytoctitle]{\glossarytitle}
4548   \GlsXtrRecordWarning{#1}
4549   \GlsXtrNoGlsWarningTail
4550 }
```

Provide some commands to accompany the record option for use with **bib2gls**.

xtrresourcefile 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.

```
4551 \newcommand*{\glstrresourcefile}[2] [] {%
```

The record option can't be set after this command.

```
4552   \disable@keys{glossaries-extra.sty}{record}%
4553   \glstr@writefields
4554   \protected@write\@auxout{\glstrresourceinit}{\string\glstr@resource{#1}{#2}}%
4555   \let\@glstr@org@see@noindex\@glstr@see@noindex
4556   \let\@glstr@see@noindex\relax
4557   \IfFileExists{#2.glstex}%
4558   {%
```

Can't scope \@input so save and restore the category code of @ to allow for internal commands in the location list.

```
4559     \edef\@bibgls@restreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4560     \makeatletter
4561     \@input{#2.glstex}%
4562     \@bibgls@restreat
4563   }%
4564   {%
4565     \GlossariesExtraWarning{No file '#2.glstex'}%
4566   }%
4567   \let\@glstr@see@noindex\@glstr@org@see@noindex
4568 }
4569 \@onlypreamble\glstrresourcefile
```

xtrresourceinit Code used during the protected write operation.

```
4570 \newcommand*{\glstrresourceinit}{}
```

trresourcecount

```
4571 \newcount\glstrresourcecount
```

trLoadResources Short cut that uses \glstrresourcefile with \jobname as the mandatory argument.

```

4572 \newcommand*{\GlsXtrLoadResources}[1][{}]{%
4573   \ifnum\glstrresourcecount=0\relax
4574     \glstrresourcefile[#1]{\jobname}%
4575   \else
4576     \glstrresourcefile[#1]{\jobname-\the\glstrresourcecount}%
4577   \fi
4578   \advance\glstrresourcecount by 1\relax
4579 }

```

glstr@resource

```

4580 \newcommand*{\glstr@resource}[2]{%

```

\glstr@fields

```

4581 \newcommand*{\glstr@fields}[1]{%

```

xtr@texencoding

```

4582 \newcommand*{\glstr@texencoding}[1]{%

```

\glstr@langtag

```

4583 \newcommand*{\glstr@langtag}[1]{%

```

@pluralsuffixes

```

4584 \newcommand*{\glstr@pluralsuffixes}[4]{%

```

tr@shortcutsval

```

4585 \newcommand*{\glstr@shortcutsval}[1]{%

```

sxtr@linkprefix

```

4586 \newcommand*{\glstr@linkprefix}[1]{%

```

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```

4587 \newcommand*{\glstr@writefields}{%

4588   \protected@write\@auxout{}%
4589     {\string\providecommand*{\string\glstr@fields}[1]{}}%
4590   \protected@write\@auxout{}%
4591     {\string\providecommand*{\string\glstr@resource}[2]{}}%
4592   \protected@write\@auxout{}%
4593     {\string\providecommand*{\string\glstr@pluralsuffixes}[4]{}}%
4594   \protected@write\@auxout{}%
4595     {\string\providecommand*{\string\glstr@shortcutsval}[1]{}}%
4596   \protected@write\@auxout{}%
4597     {\string\providecommand*{\string\glstr@linkprefix}[1]{}}%
4598   \protected@write\@auxout{}{\string\glstr@fields{\@gls@keymap}}%

4599   \protected@write\@auxout{}%
4600     {\string\providecommand*{\string\glstr@record}[5]{}}%

```

If any languages have been loaded, the language tag will be available in `\CurrentTrackedLanguageTag` (provided by `tracklang`). For multilingual documents, the required locale will have to be indicated in the sort key when using `\glxtrresourcefile`.

```

4601 \ifdef\CurrentTrackedLanguageTag
4602 {%
4603   \protected@write\@auxout{}\{%
4604     \string\glxtr@langtag{\CurrentTrackedLanguageTag}}%
4605   }%
4606   {%
4607     \protected@write\@auxout{}\{\string\glxtr@pluralsuffixes
4608       {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
4609       {\glxtrabbrvpluralsuffix}}%
4610   \ifdef\inputencodingname
4611     {%
4612       \protected@write\@auxout{}\{\string\glxtr@texencoding{\inputencodingname}}%
4613     }%
4614     {%

```

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.)

```

4615   \@ifpackageloaded{fontspec}%
4616   {\protected@write\@auxout{}\{\string\glxtr@texencoding{utf8}}}%
4617   }%
4618 }%
4619 \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.

```

4620 \AtBeginDocument
4621   {\protected@write\@auxout{}\{\string\glxtr@linkprefix{\glolinkprefix}}}%
4622   \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.

```

4623 \ifglautomake
4624   \IfFileExists{\jobname.aux}%
4625   {\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`.

```

4626   \ifx\@gls@doautomake\@gls@doautomake@err
4627     \let\@gls@doautomake\relax
4628   \fi
4629 \fi
4630 }

```

`do@automake@err`

```

4631 \newcommand*{\@gls@doautomake@err}{%
4632   \PackageError{glossaries}{You must use

```



```

4633 \string\makeglossaries\space with automake=true}
4634 {%
4635     Either remove the automake=true setting or
4636     add \string\makeglossaries\space to your document preamble.%
4637 }%
4638 }

```

Allow locations specific to a particular counter to be recorded.

\glxtr@record

```

4639 \newcommand*{\glxtr@record}[5]{}

```

r@counterrecord Aux file command.

```

4640 \newcommand*{\glxtr@counterrecord}[3]{%
4641     \glxtrfieldlistgadd{#1}{record.#2}{#3}%
4642 }

```

unterrecordhook Hook used by \@glxtr@dorecord.

```

4643 \newcommand*{\@glxtr@counterrecordhook}{}

```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```

4644 \newcommand*{\GlsXtrRecordCounter}[1]{%
4645     \@glxtr@recordcounter{#1}%
4646 }
4647 \@onlypreamble\GlsXtrRecordCounter

```

docounterrecord

```

4648 \newcommand*{\@glxtr@docounterrecord}[1]{%
4649     \protected@write\@auxout{}\string\glxtr@counterrecord
4650         {\@glxtr@label}{#1}{\csuse{the#1}}}%
4651 }

```

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.

```

4652 \newcommand*{\glxtrglossentry}[1]{%
4653     \glxtrtitleorpdforheading
4654     {\@glxtrglossentry{#1}}%
4655     {\glsentryname{#1}}%
4656     {\glxtrheadname{#1}}%
4657 }

```

lsxtrglossentry Another test is needed in case \@glxtrglossentry has been written to the table of contents.

```

4658 \newrobustcmd*{\@glxtrglossentry}[1]{%

```

```

4659 \glxtrtitleorpdforheading
4660 {%
4661   \glsdoifexists{#1}%
4662   {%
4663     \begingroup
4664     \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4665     \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4666     \ifglshasparent{#1}%
4667     {\GlsXtrStandaloneSubEntryItem{#1}}%
4668     {\glseentryitem{#1}}%
4669     \glstarget{#1}{\glossentryname{#1}}%
4670   \endgroup
4671   }%
4672 }%
4673 {\glseentryname{#1}}%
4674 {\glxtrheadname{#1}}%
4675 }

```

**oneGlossaryType** To make it easier to adjust the definition of `\currentglossary` within `\glxtrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```

4676 \newcommand{\GlsXtrStandaloneGlossaryType}{\glseentrytype{\glscurrententrylabel}}

```

**oneSubEntryItem** Used for sub-entries in standalone format. The argument is the entry's label.

```

4677 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
4678   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}}%
4679 }

```

**glossentryother** As `\glxtrglossentry` but uses a different field. First argument is code to use in the header. The second argument is the entry's label. The third argument is the internal field label. This needs to be expandable in case it occurs in a sectioning command so it can't have an optional argument.

```

4680 \newcommand*{\glxtrglossentryother}[3]{%
4681   \ifstrempy{#1}%
4682   {%
4683     \ifcsdef{glxtrhead#3}%
4684     {%
4685       \glxtrtitleorpdforheading
4686       {\@glxtrglossentryother{#2}{#3}{#1}}%
4687       {\@gls@entry@field{#2}{#3}}%
4688       {\csuse{glxtrhead#3}{#2}}%
4689     }%
4690   }%
4691   \glxtrtitleorpdforheading
4692   {\@glxtrglossentryother{#2}{#3}{#1}}%
4693   {\@gls@entry@field{#2}{#3}}%
4694   {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
4695 }%

```

```

4696 }%
4697 {%
4698   \glxstrtitleorpdforheading
4699   {\@glxstrglossentryother{#2}{#3}{#1}}%
4700   {\@glx@entry@field{#2}{#3}}%
4701   {#1}%
4702 }%
4703 }

```

`glossentryother` As `\@glxstrglossentry` but uses a different field.

```

4704 \newrobustcmd*{\@glxstrglossentryother}[3]{%
4705   \glxstrtitleorpdforheading
4706   {%
4707     \glxdoifexists{#1}%
4708     {%
4709       \begingroup
4710       \edef\glxcurrententrylabel{\glxdetoklabel{#1}}%
4711       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
4712       \ifglshasparent{#1}%
4713       {\GlsXtrStandaloneSubEntryItem{#1}}%
4714       {\glxentryitem{#1}}%
4715       \glxtarget{#1}{\glossentrynameother{#1}{#2}}%
4716     \endgroup
4717   }%
4718 }%
4719 {\@glx@entry@field{#1}{#2}}%
4720 {#3}%
4721 }

```

`ntunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```

4722 \newcommand*{\printunsrtglossary}{%
4723   \@ifstar\s@printunsrtglossary\@printunsrtglossary
4724 }

```

`ntunsrtglossary` Unstarred version.

```

4725 \newcommand*{\@printunsrtglossary}[1][ ]{%
4726   \@printglossary{type=\glxdefaulttype,#1}{\@print@unsrt@glossary}%
4727 }

```

`ntunsrtglossary` Starred version.

```

4728 \newcommand*{\s@printunsrtglossary}[2][ ]{%
4729   \begingroup
4730   #2%
4731   \@printglossary{type=\glxdefaulttype,#1}{\@print@unsrt@glossary}%
4732   \endgroup
4733 }

```

`\unsrtglossaries` Similar to `\printnoidxglossaries` but it displays all entries defined for the given glossary without sorting.

```
4734 \newcommand*{\printunsrtglossaries}{%
4735   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
4736 }
```

`@unsrt@glossary`

```
4737 \newcommand*{\@print@unsrt@glossary}{%
4738   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4739   \glossarypreamble

   check for empty list
4740   \glstrifemptyglossary{\@glo@type}%
4741   {%
4742     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
4743   }%
4744   {%

4745     \key@ifundefined{glossentry}{group}%
4746     {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
4747     {\let\@gls@getgrouptitle\@glstr@unsrt@getgrouptitle}%
4748     \def\@gls@currentlettergroup{}}%
```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```
4749   \def\@glstr@doglossary{%
4750     \begin{theglossary}%
4751     \glossaryheader
4752     \glresetentrylist
4753   }%
4754   \expandafter\@for\expandafter\glscurrententrylabel\expandafter
4755     :\expandafter=\csname glo@list@\@glo@type\endcsname\do{%
4756     \ifdefempty{\glscurrententrylabel}
4757     }%
4758   }%
```

Provide a hook (for example to measure width).

```
4759     \let\glstr@process\@firstofone
4760     \let\printunsrtglossaryskipentry
4761         \glstr@printunsrtglossaryskipentry
4762     \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%
```

Don't check group for child entries.

```
4763     \glstr@process
4764     {%
4765       \ifglshasparent{\glscurrententrylabel}{}%
4766       {%
4767         \@glstr@checkgroup\glscurrententrylabel
4768         \expandafter\appto\expandafter\@glstr@doglossary\expandafter
4769           {\@glstr@groupheading}%
4770       }%
```

```

4771      \eappto\@glxtr@doglossary{%
4772      \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
4773      }%
4774      }%
4775      }%
4776      \appto\@glxtr@doglossary{\end{theglossary}}%
4777      \printunsrtglossarypredoglossary
4778      \@glxtr@doglossary
4779      }%
4780      \glossarypostamble
4781      }

```

entryprocesshook

```

4782 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{}

```

glossaryskipentry

```

4783 \newcommand*{\printunsrtglossaryskipentry}{%
4784   \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
4785   can only be used within \string\printunsrtglossaryentryprocesshook}{}%
4786 }

```

entryprocesshook

```

4787 \newcommand*{\@glxtr@printunsrtglossaryskipentry}{%
4788   \let\glxtr@process\@gobble
4789 }

```

glossarypredoglossary

```

4790 \newcommand*{\printunsrtglossarypredoglossary}{}

```

glossary@handler

```

4791 \newcommand{\@printunsrt@glossary@handler}[1]{%
4792   \xdef\glscurrententrylabel{#1}%
4793   \printunsrtglossaryhandler\glscurrententrylabel
4794 }

```

glossaryhandler

```

4795 \newcommand{\printunsrtglossaryhandler}[1]{%
4796   \glxtrunsrtdo{#1}%
4797 }

```

glxtriflabelinlist

```

\glxtriflabelinlist{<label>}{<list>}{<true>}{<false>}

```

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of `\@glx@ifinlist` which ensures the label and list are fully expanded.

```

4798 \newrobustcmd*{\glxtriflabelinlist}[4]{%
4799   \protected@edef\@glxtr@doiflabelinlist{\noexpand\@glx@ifinlist{#1}{#2}}%
4800   \@glxtr@doiflabelinlist{#3}{#4}%
4801 }

```

srtglossaryunit

```

4802 \newcommand{\print@op@unsrtglossaryunit}[2][{}]{%
4803   \s@printunsrtglossary[type=\glstypedefaulttype,#1]{%
4804     \printunsrtglossaryunitsetup{#2}%
4805   }%
4806 }

```

ossaryunitsetup

```

4807 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
4808   \renewcommand{\printunsrtglossaryhandler}[1]{%
4809     \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
4810     {\glxtrunsrtdo{##1}}%
4811   }%
4812 }%

```

Only the target names should have the prefixes adjusted as \glx etc need the original \glolinkprefix. The \@gobble part discards \glolinkprefix.

```

4813 \ifcsundef{theH#1}%
4814 {%
4815   \renewcommand*{\@glxtr@hypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
4816 }%
4817 {%
4818   \renewcommand*{\@glxtr@hypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
4819 }%
4820 \renewcommand*{\glossarysection}[2][{}]{%
4821   \appto\glossarypostamble{\glspare\medskip\glspare}%
4822 }

```

srtglossaryunit

```

4823 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]{%
4824   \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
4825     requires the record=only or record=alsoindex package option}{}%
4826 }

```

t@getgrouptitle

```

4827 \newrobustcmd*{\@glxtr@unsrt@getgrouptitle}[2]{%
4828   \protected@edef\@glxtr@titlelabel{\glxtr@grouptitle@#1}%
4829   \@onelevel@sanitize\@glxtr@titlelabel
4830   \ifcsdef{\@glxtr@titlelabel}
4831   {\letcs{#2}{\@glxtr@titlelabel}}%
4832   {\def#2{#1}}%
4833 }

```

\glxtrunsrtdo Provide a user-level call to \@glxtr@noidx@do to make it easier to define a new handler.

```

4834 \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).

```
4835 \newcommand*{\glsxtrgroupfield}{group}
```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while `\@glsxtr@doglossary` is being constructed rather than in the handler.

`lsxtr@checkgroup` The argument is the entry's label. (This block of code was formerly in `\@glsxtr@noidx@do`.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in `\@glsxtr@groupheading`, which will be empty if no heading is required.

```
4836 \newcommand*{\@glsxtr@checkgroup}[1]{%
4837   \def\@glsxtr@groupheading{}%
4838   \key@ifundefined{glossentry}{group}%
4839   {%
4840     \letcs{\@gls@sort}{glo\@glsdetoklabel{#1}@sort}%
4841     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
4842   }%
4843   {%
4844     \protected@edef\@glo@thislettergrp{%
4845       \csuse{glo\@glsdetoklabel{#1}\@glsxtrgroupfield}}%
4846     }%
4847     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
4848     {}%
4849     {%
4850       \ifdefempty{\@gls@currentlettergroup}{}%
4851       {\def\@glsxtr@groupheading{\gls@groupskip}}%
4852       \eappto\@glsxtr@groupheading{%
4853         \noexpand\gls@groupheading{\expandonce\@glo@thislettergrp}%
4854       }%
4855     }%
4856     \let\@gls@currentlettergroup\@glo@thislettergrp
4857 }
```

`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.

```
4858 \newcommand{\@glsxtr@noidx@do}[1]{%
4859   \ifglssentryexists{#1}%
4860   {%
4861     \global\letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4862     \global\letcs{\@gls@location}{glo\@glsdetoklabel{#1}@location}%
4863     \ifglshasparent{#1}%
4864     {%
4865       \gls@level=\csuse{glo\@glsdetoklabel{#1}@level}\relax
4866       \ifdefvoid{\@gls@location}%

```

```

4867     {%
4868     \ifdefvoid{\@gls@loclist}%
4869     {%
4870     \subglossentry{\gls@level}{#1}{}%
4871     }%
4872     {%
4873     \subglossentry{\gls@level}{#1}%
4874     {%
4875     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4876     }%
4877     }%
4878     }%
4879     {%
4880     \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
4881     }%
4882     }%
4883     {%

4884     \ifdefvoid{\@gls@location}%
4885     {%
4886     \ifdefvoid{\@gls@loclist}
4887     {%
4888     \glossentry{#1}{}%
4889     }%
4890     {%
4891     \glossentry{#1}%
4892     {%
4893     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4894     }%
4895     }%
4896     }%
4897     {%
4898     \glossentry{#1}%
4899     {%
4900     \glossaryentrynumbers{\@gls@location}%
4901     }%
4902     }%
4903     }%
4904     }%
4905     {}%
4906 }

```

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`

```

4907 \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>}`

```

4908 \newcommand*{\@glsxtrnewgls}[4]{%
4909   \ifdef{#3}%
4910   {%
4911     \PackageError{glossaries-extra}{Command \string#3\space already
4912 defined}{}%
4913   }%
4914   {%
4915     \ifcsdef{@#4like@#2}%
4916     {%
4917       \advance\@glsxtrnewgls@inner by \@ne
4918       \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
4919     }%
4920     {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
4921     \expandafter\newrobustcmd\expandafter*\expandafter
4922       #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
4923     \ifstrempy{#1}%
4924     {%
4925       \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][ ]{%
4926         \new@ifnextchar[%
4927           {\csname @#4@\endcsname{##1}{#2##2}}%
4928           {\csname @#4@\endcsname{##1}{#2##2} [ ]}%
4929         }%
4930       }%
4931     }%
4932     \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][ ]{%
4933       \new@ifnextchar[%
4934         {\csname @#4@\endcsname{#1,##1}{#2##2}}%
4935         {\csname @#4@\endcsname{#1,##1}{#2##2} [ ]}%
4936       }%
4937     }%
4938   }%
4939 }

```

`\glsxtrnewgls` `\glsxtrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```

4940 \newrobustcmd*{\glsxtrnewgls}[3][ ]{%
4941   \@glsxtrnewgls{#1}{#2}{#3}{gls}%
4942 }

```

`\lsxtrnewglslike` Provide a way to conveniently define commands that behave like `\gls`, `\glspl`, `\Gls` and `\Glspl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
4943 \newrobustcmd*{\glsxtrnewglslike}[6] [] {%
4944   \@glsxtrnewgls{#1}{#2}{#3}{gls}%
4945   \@glsxtrnewgls{#1}{#2}{#4}{glspl}%
4946   \@glsxtrnewgls{#1}{#2}{#5}{Gls}%
4947   \@glsxtrnewgls{#1}{#2}{#6}{Glspl}%
4948 }
```

`\lsxtrnewGLSlike` Provide a way to conveniently define commands that behave like `\GLS`, `\GLSpl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
4949 \newrobustcmd*{\glsxtrnewGLSlike}[4] [] {%
4950   \@glsxtrnewgls{#1}{#2}{#3}{GLS}%
4951   \@glsxtrnewgls{#1}{#2}{#4}{GLSpl}%
4952 }
```

`\glsxtrnewrgls` As `\glsxtrnewgls` but for `\rgls`.

```
4953 \newrobustcmd*{\glsxtrnewrgls}[3] [] {%
4954   \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
4955 }
```

`\sxtrnewrglslike` As `\glsxtrnewglslike` but for `\rgls` etc.

```
4956 \newrobustcmd*{\glsxtrnewrglslike}[6] [] {%
4957   \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
4958   \@glsxtrnewgls{#1}{#2}{#4}{rglspl}%
4959   \@glsxtrnewgls{#1}{#2}{#5}{rGls}%
4960   \@glsxtrnewgls{#1}{#2}{#6}{rGlspl}%
4961 }
```

`\sxtrnewrGLSlike` As `\glsxtrnewGLSlike` but for `\rGLS` etc.

```
4962 \newrobustcmd*{\glsxtrnewrGLSlike}[4] [] {%
4963   \@glsxtrnewgls{#1}{#2}{#3}{rGLS}%
4964   \@glsxtrnewgls{#1}{#2}{#4}{rGLSpl}%
4965 }
```

Provide easy access to record count fields.

`\totalRecordCount` Access total record count. This is designed to be expandable. The argument is the label.

```
4966 \newcommand*{\GlsXtrTotalRecordCount}[1] {%
4967   \ifcsdef{glo@glstoklabel{#1}@recordcount}%
4968   {\csname glo@glstoklabel{#1}@recordcount\endcsname}%
4969   {0}%
4970 }
```

`\sXtrRecordCount` Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
4971 \newcommand*{\GlsXtrRecordCount}[2] {%
```

```

4972 \ifcsdef{glo@\glxstrdetoklabel{#1}@recordcount.#2}%
4973 {\csname glo@\glxstrdetoklabel{#1}@recordcount.#2\endcsname}%
4974 {0}%
4975 }

```

**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.

```

4976 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
4977 \ifcsdef{glo@\glxstrdetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}}%
4978 {\csname glo@\glxstrdetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}\endcsname}%
4979 {0}%
4980 }

```

**trdetoklocation**

```

4981 \newcommand*{\glxstrdetoklocation}[1]{#1}

```

**ablerecordcount**

```

4982 \newcommand*{\glxstreablerecordcount}{%
4983 \renewcommand*{\gls}{\rgls}%
4984 \renewcommand*{\Gls}{\rGls}%
4985 \renewcommand*{\glspl}{\rglspl}%
4986 \renewcommand*{\Glspl}{\rGlspl}%
4987 \renewcommand*{\GLS}{\rGLS}%
4988 \renewcommand*{\GLSpl}{\rGLSpl}%
4989 }

```

**ordtriggervalue** The value used by the record trigger test. The argument is the entry's label.

```

4990 \newcommand*{\glxstrrecordtriggervalue}[1]{%
4991 \GlsXtrTotalRecordCount{#1}%
4992 }

```

**dCountAttribute**

```

4993 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
4994 \@for\@glxstr@cat:=#1\do
4995 {%
4996 \ifdefempty{\@glxstr@cat}{}%
4997 {%
4998 \glsssetcategoryattribute{\@glxstr@cat}{recordcount}{#2}%
4999 }%
5000 }%
5001 }

```

**rifrecordtrigger**

```
\glxstrifrecordtrigger{<label>}{<trigger format>}{<normal>}
```

```

5002 \newcommand*{\glxtrifrecordtrigger}[3]{%
5003   \glshasattribute{#1}{recordcount}%
5004   {%
5005     \ifnum\glxtrrecordtriggervalue{#1}>\glsggetattribute{#1}{recordcount}\relax
5006     #3%
5007     \else
5008     #2%
5009     \fi
5010   }%
5011   {#3}%
5012 }

```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```

5013 \newcommand*{\@glxtr@rglstrigger@record}[3]{%
5014   \edef\glslabel{\glsdetoklabel{#2}}%
5015   \let\@gls@link@label\glslabel
5016   \def\@glxtr@thevalue{%
5017     \def\@glxtr@theHvalue{\@glxtr@thevalue}%
5018     \def\@glsnumberformat{glstriggerrecordformat}%
5019     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
5020     \edef\glstype{\csname glo@\glslabel @type\endcsname}%
5021     \def\@glxtr@thevalue{%
5022       \def\@glxtr@theHvalue{\@glxtr@thevalue}%
5023       \glxtrinitwrgloss
5024       \glslinkpresetkeys
5025       \setkeys{glslink}{#1}%
5026       \glslinkpostsetkeys
5027       \ifdefempty{\@glxtr@thevalue}%
5028       {%
5029         \@gls@saveentrycounter
5030       }%
5031       {%
5032         \let\theglentrycounter\@glxtr@thevalue
5033         \def\theHglentrycounter{\@glxtr@theHvalue}%
5034       }%
5035       \ifglxtrinitwrglossbefore
5036         \@do@wrglossary{#2}%
5037       \fi
5038       #3%
5039       \ifglxtrinitwrglossbefore
5040       \else
5041         \@do@wrglossary{#2}%
5042       \fi
5043       \ifKV@glslink@local
5044         \glslocalunset{#2}%
5045       \else
5046         \glsunset{#2}%
5047       \fi

```

```

5048 }

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by
bib2gls.
5049 \newcommand*{\glstriggerrecordformat}[1]{

\rgls
5050 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

\@rgls
5051 \newcommand*{\@rgls}[2][]{%
5052   \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5053 }

\@rgls@
5054 \def\@rgls@#1#2[#3]{%
5055   \glstrifrecordtrigger{#2}%
5056   {%
5057     \@glstr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5058   }%
5059   {%
5060     \@gls@{#1}{#2}[#3]%
5061   }%
5062 }%

\rglspl
5063 \newrobustcmd*{\rglspl}{\@gls@hyp@opt\@rglspl}

\@rglspl
5064 \newcommand*{\@rglspl}[2][]{%
5065   \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
5066 }

\@rglspl@
5067 \def\@rglspl@#1#2[#3]{%
5068   \glstrifrecordtrigger{#2}%
5069   {%
5070     \@glstr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
5071   }%
5072   {%
5073     \@glspl@{#1}{#2}[#3]%
5074   }%
5075 }%

\rGls
5076 \newrobustcmd*{\rGls}{\@gls@hyp@opt\@rGls}

```

```

\@rGls
5077 \newcommand*{\@rGls}[2][{}]{%
5078   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5079 }

\@rGls@
5080 \def\@rGls@#1#2[#3]{%
5081   \glstrifrecordtrigger{#2}%
5082   {%
5083     \@glstr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5084   }%
5085   {%
5086     \@Gls@{#1}{#2}[#3]%
5087   }%
5088 }%

\rGlspl
5089 \newrobustcmd*{\rGlspl}{\@glshyp@opt\rGlspl}

\@rGlspl
5090 \newcommand*{\@rGlspl}[2][{}]{%
5091   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
5092 }

\@rGlspl@
5093 \def\@rGlspl@#1#2[#3]{%
5094   \glstrifrecordtrigger{#2}%
5095   {%
5096     \@glstr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5097   }%
5098   {%
5099     \@Glspl@{#1}{#2}[#3]%
5100   }%
5101 }%

\rGLS
5102 \newrobustcmd*{\rGLS}{\@glshyp@opt\rGLS}

\@rGLS
5103 \newcommand*{\@rGLS}[2][{}]{%
5104   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}[]}%
5105 }

\@rGLS@
5106 \def\@rGLS@#1#2[#3]{%
5107   \glstrifrecordtrigger{#2}%
5108   {%
5109     \@glstr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%

```

```

5110 }%
5111 {%
5112   \@GLS@{#1}{#2}[#3]%
5113 }%
5114 }%

```

\rGLSp1

```

5115 \newrobustcmd*{\rGLSp1}{\@gls@hyp@opt\rGLSp1}

```

\@rGLSp1

```

5116 \newcommand*{\@rGLSp1}[2][{}]{%
5117   \new@ifnextchar[{\@rGLSp1@{#1}{#2}}{\@rGLSp1@{#1}{#2}[{}]}%
5118 }

```

\@rGLSp1@

```

5119 \def\@rGLSp1@#1#2[#3]{%
5120   \glsxtrifrecordtrigger{#2}%
5121   {%
5122     \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
5123   }%
5124   {%
5125     \@GLSp1@{#1}{#2}[#3]%
5126   }%
5127 }%

```

\rglsformat

```

5128 \newcommand*{\rglsformat}[2]{%
5129   \glsifregular{#1}
5130   {\glsentryfirst{#1}}%
5131   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5132 }

```

\rglsplformat

```

5133 \newcommand*{\rglsplformat}[2]{%
5134   \glsifregular{#1}
5135   {\glsentryfirstplural{#1}}%
5136   {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5137 }

```

\rGlsformat

```

5138 \newcommand*{\rGlsformat}[2]{%
5139   \glsifregular{#1}
5140   {\Glsentryfirst{#1}}%
5141   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5142 }

```

\rGlsplformat

```

5143 \newcommand*{\rGlsplformat}[2]{%

```

```

5144 \glsifregular{#1}
5145 {\Glsentryfirstplural{#1}}%
5146 {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
5147 }

```

\rGLSformat

```

5148 \newcommand*{\rGLSformat}[2]{%
5149 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSformat{#1}{#2}}%
5150 }

```

\rGLSplformat

```

5151 \newcommand*{\rGLSplformat}[2]{%
5152 \expandafter\mfirstucMakeUppercase\expandafter{\rGLSplformat{#1}{#2}}%
5153 }

```

## 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.

```

5154 \newcommand{\@glsxtr@do@inc@linkcount}{%
    Does this entry have the linkcount attribute set?
5155 \glsifattribute{\glslabel}{linkcount}{true}%
5156 {%
    Does the counter exist?
5157 \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
5158 {%
    Counter doesn't exist, so define it.
5159 \newcounter{glsxtr@linkcount@\glslabel}%
    If linkcountmaster is set, add to counter reset.
5160 \glshasattribute{\glslabel}{linkcountmaster}%
5161 {%
    Need to ensure values are fully expanded.
5162 \begingroup
5163 \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@\glslabel}%
5164 {\glsgetattribute{\glslabel}{linkcountmaster}}}%
5165 \x
5166 }%

```



```

5167     {}%
5168   }%
    Increment counter:
5169   \glxstrinclinkcounter{glxstr@linkcount@glslabel}%
5170 }%
5171 {}%
5172 }

rlinkcounter   May be redefined to use \refstepcounter if required.
5173 \newcommand*{\glxstrinclinkcounter}[1]{\stepcounter{#1}}

linkCounterValue   Expands to the associated link counter register or 0 if not defined.
5174 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
5175   \ifcsundef{c@glxstr@linkcount@#1}{0}{\csname c@glxstr@linkcount@#1\endcsname}%
5176 }

rTheLinkCounter   Expands to the display value of the associated link counter or 0 if not defined.
5177 \newcommand*{\GlsXtrTheLinkCounter}[1]{%
5178   \ifcsundef{theglsxtr@linkcount@#1}{0}%
5179   {\csname theglxstr@linkcount@#1\endcsname}%
5180 }

fLinkCounterDef   Tests if the counter has been defined
5181 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
5182   \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
5183 }

LinkCounterName   Expands to the associated link counter name. (No check for existence.)
5184 \newcommand*{\GlsXtrLinkCounterName}[1]{glxstr@linkcount@#1}

ableLinkCounting   \GlsXtrEnableLinkCounting[master counter]{categories}

    Enable link counting for the given categories.
5185 \newcommand*{\GlsXtrEnableLinkCounting}[2][{}]{%
5186   \let\glxstr@inc@linkcount\@glxstr@do@inc@linkcount
5187   \@for\@glxstr@label:=#2\do
5188   {%
5189     \glsssetcategoryattribute{\@glxstr@label}{linkcount}{true}%
5190     \ifstrempy{#1}{}%
5191     {%
5192       \ifcsundef{c@#1}%
5193       {\@nocounterr{#1}}%
5194       {\glsssetcategoryattribute{\@glxstr@label}{linkcountmaster}{#1}}%
5195     }%
5196   }%
5197 }
5198 \@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.

```
5199 \@ifpackageloaded{glossaries-accsupp}
5200 {
```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```
5201 \newcommand*{\glsaccessname}[1]{%
5202 \glsnameaccessdisplay
5203 {%
5204 \glsentryname{#1}%
5205 }%
5206 {#1}%
5207 }
```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```
5208 \newcommand*{\Glsaccessname}[1]{%
5209 \glsnameaccessdisplay
5210 {%
5211 \Glsentryname{#1}%
5212 }%
5213 {#1}%
5214 }
```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```
5215 \newcommand*{\GLSaccessname}[1]{%
5216 \glsnameaccessdisplay
5217 {%
5218 \mfirstucMakeUppercase{\glsentryname{#1}}%
5219 }%
5220 {#1}%
5221 }
```

`\glsaccesstext` Display the text value (no link and no check for existence).

```
5222 \newcommand*{\glsaccesstext}[1]{%
5223 \glstextaccessdisplay
5224 {%
5225 \glsentrytext{#1}%
5226 }%
5227 {#1}%
5228 }
```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```

5229 \newcommand*{\Glsaccesstext}[1]{%
5230   \glstextaccessdisplay
5231   {%
5232     \Glsentrytext{#1}%
5233   }%
5234   {#1}%
5235 }
```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```

5236 \newcommand*{\GLSaccesstext}[1]{%
5237   \glstextaccessdisplay
5238   {%
5239     \mfirstucMakeUppercase{\Glsentrytext{#1}}%
5240   }%
5241   {#1}%
5242 }
```

`glsaccessplural` Display the plural value (no link and no check for existence).

```

5243 \newcommand*{\glsaccessplural}[1]{%
5244   \glspluralaccessdisplay
5245   {%
5246     \Glsentryplural{#1}%
5247   }%
5248   {#1}%
5249 }
```

`GLSaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

5250 \newcommand*{\GLSaccessplural}[1]{%
5251   \glspluralaccessdisplay
5252   {%
5253     \Glsentryplural{#1}%
5254   }%
5255   {#1}%
5256 }
```

`GLSaccessplural` Display the plural value (no link and no check for existence) converted to upper case.

```

5257 \newcommand*{\GLSaccessplural}[1]{%
5258   \glspluralaccessdisplay
5259   {%
5260     \mfirstucMakeUppercase{\Glsentryplural{#1}}%
5261   }%
5262   {#1}%
5263 }
```

`\glsaccessfirst` Display the first value (no link and no check for existence).

```

5264 \newcommand*{\glsaccessfirst}[1]{%
5265   \glsfirstaccessdisplay
5266   {%
5267     \glstryfirst{#1}%
5268   }%
5269   {#1}%
5270 }

```

`\Glsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

5271 \newcommand*{\Glsaccessfirst}[1]{%
5272   \glsfirstaccessdisplay
5273   {%
5274     \Glsentryfirst{#1}%
5275   }%
5276   {#1}%
5277 }

```

`\GLSaccessfirst` Display the first value (no link and no check for existence) converted to upper case.

```

5278 \newcommand*{\GLSaccessfirst}[1]{%
5279   \glsfirstaccessdisplay
5280   {%
5281     \mfirstucMakeUppercase{\glstryfirst{#1}}%
5282   }%
5283   {#1}%
5284 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```

5285 \newcommand*{\glsaccessfirstplural}[1]{%
5286   \glsfirstpluralaccessdisplay
5287   {%
5288     \glstryfirstplural{#1}%
5289   }%
5290   {#1}%
5291 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

5292 \newcommand*{\Glsaccessfirstplural}[1]{%
5293   \glsfirstpluralaccessdisplay
5294   {%
5295     \Glsentryfirstplural{#1}%
5296   }%
5297   {#1}%
5298 }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) converted to upper case.

```

5299 \newcommand*{\GLSaccessfirstplural}[1]{%

```

```

5300   \glsfirstpluralaccessdisplay
5301   {%
5302       \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
5303   }%
5304   {#1}%
5305   }

```

**glsaccesssymbol** Display the symbol value (no link and no check for existence).

```

5306   \newcommand*{\glsaccesssymbol}[1]{%
5307       \glssymbolaccessdisplay
5308       {%
5309           \glsentrysymbol{#1}%
5310       }%
5311       {#1}%
5312   }

```

**Glsaccesssymbol** Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

5313   \newcommand*{\Glsaccesssymbol}[1]{%
5314       \glssymbolaccessdisplay
5315       {%
5316           \Glsentrysymbol{#1}%
5317       }%
5318       {#1}%
5319   }

```

**GLSaccesssymbol** Display the symbol value (no link and no check for existence) converted to upper case.

```

5320   \newcommand*{\GLSaccesssymbol}[1]{%
5321       \glssymbolaccessdisplay
5322       {%
5323           \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
5324       }%
5325       {#1}%
5326   }

```

**esssymbolplural** Display the symbolplural value (no link and no check for existence).

```

5327   \newcommand*{\glsaccesssymbolplural}[1]{%
5328       \glssymbolpluralaccessdisplay
5329       {%
5330           \glsentrysymbolplural{#1}%
5331       }%
5332       {#1}%
5333   }

```

**esssymbolplural** Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

5334   \newcommand*{\Glsaccesssymbolplural}[1]{%
5335       \glssymbolpluralaccessdisplay

```

```

5336    {%
5337        \Glsentrysymbolplural{#1}%
5338    }%
5339    {#1}%
5340 }

```

**esssymbolplural** Display the symbolplural value (no link and no check for existence) converted to upper case.

```

5341 \newcommand*{\GLSaccesssymbolplural}[1]{%
5342     \glssymbolpluralaccessdisplay
5343     {%
5344         \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%
5345     }%
5346     {#1}%
5347 }

```

**\glsaccessdesc** Display the desc value (no link and no check for existence).

```

5348 \newcommand*{\glsaccessdesc}[1]{%
5349     \glsdescriptionaccessdisplay
5350     {%
5351         \glentrydesc{#1}%
5352     }%
5353     {#1}%
5354 }

```

**\Glsaccessdesc** Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

5355 \newcommand*{\Glsaccessdesc}[1]{%
5356     \glsdescriptionaccessdisplay
5357     {%
5358         \Glsentrydesc{#1}%
5359     }%
5360     {#1}%
5361 }

```

**\GLSaccessdesc** Display the desc value (no link and no check for existence) converted to upper case.

```

5362 \newcommand*{\GLSaccessdesc}[1]{%
5363     \glsdescriptionaccessdisplay
5364     {%
5365         \mfirstucMakeUppercase{\glentrydesc{#1}}%
5366     }%
5367     {#1}%
5368 }

```

**ccessdescplural** Display the descplural value (no link and no check for existence).

```

5369 \newcommand*{\glsaccessdescplural}[1]{%
5370     \glsdescriptionpluralaccessdisplay
5371     {%
5372         \glentrydescplural{#1}%

```

```

5373     }%
5374     {#1}%
5375 }

```

`\accessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

5376 \newcommand*{\Glsaccessdescplural}[1]{%
5377   \glsdescriptionpluralaccessdisplay
5378   {%
5379     \Glsentrydescplural{#1}%
5380   }%
5381   {#1}%
5382 }

```

`\accessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

5383 \newcommand*{\GLSaccessdescplural}[1]{%
5384   \glsdescriptionpluralaccessdisplay
5385   {%
5386     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5387   }%
5388   {#1}%
5389 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

5390 \newcommand*{\glsaccessshort}[1]{%
5391   \glsshortaccessdisplay
5392   {%
5393     \glsentryshort{#1}%
5394   }%
5395   {#1}%
5396 }

```

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5397 \newcommand*{\Glsaccessshort}[1]{%
5398   \glsshortaccessdisplay
5399   {%
5400     \Glsentryshort{#1}%
5401   }%
5402   {#1}%
5403 }

```

`\GLSAccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

5404 \newcommand*{\GLSAccessshort}[1]{%
5405   \glsshortaccessdisplay
5406   {%
5407     \mfirstucMakeUppercase{\glsentryshort{#1}}%
5408   }%

```

```

5409     {#1}%
5410 }

```

`\lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

5411 \newcommand*{\lsaccessshortpl}[1]{%
5412   \glshortpluralaccessdisplay
5413   {%
5414     \glentryshortpl{#1}%
5415   }%
5416   {#1}%
5417 }

```

`\Lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5418 \newcommand*{\Lsaccessshortpl}[1]{%
5419   \glshortpluralaccessdisplay
5420   {%
5421     \Glentryshortpl{#1}%
5422   }%
5423   {#1}%
5424 }

```

`\LSaccessshortpl` Display the shortplural value (no link and no check for existence) converted to upper case.

```

5425 \newcommand*{\LSaccessshortpl}[1]{%
5426   \glshortpluralaccessdisplay
5427   {%
5428     \mfirstucMakeUppercase{\glentryshortpl{#1}}%
5429   }%
5430   {#1}%
5431 }

```

`\glsaccesslong` Display the long form (no link and no check for existence).

```

5432 \newcommand*{\glsaccesslong}[1]{%
5433   \glslongaccessdisplay{\glentrylong{#1}}{#1}%
5434 }

```

`\Glsaccesslong` Display the long form (no link and no check for existence).

```

5435
5436 \newcommand*{\Glsaccesslong}[1]{%
5437   \glslongaccessdisplay{\Glentrylong{#1}}{#1}%
5438 }

```

`\GLSaccesslong` Display the long value (no link and no check for existence) converted to upper case.

```

5439 \newcommand*{\GLSaccesslong}[1]{%
5440   \glslongaccessdisplay
5441   {%
5442     \mfirstucMakeUppercase{\glentrylong{#1}}%
5443   }%

```



```

5444     {#1}%
5445 }

glsaccesslongpl Display the long plural form (no link and no check for existence).
5446 \newcommand*{\glsaccesslongpl}[1]{%
5447   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
5448 }

Glsaccesslongpl Display the long plural form (no link and no check for existence).
5449
5450 \newcommand*{\Glsaccesslongpl}[1]{%
5451   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5452 }

GLSaccesslongpl Display the longplural value (no link and no check for existence) converted to upper case.
5453 \newcommand*{\GLSaccesslongpl}[1]{%
5454   \glslongpluralaccessdisplay
5455   {%
5456     \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5457   }%
5458   {#1}%
5459 }

Keys for accessibility support.
5460 \define@key{glsxtrabrv}{access}{%
5461   \def\@gls@nameaccess{#1}%
5462 }
5463 \define@key{glsxtrabrv}{textaccess}{%
5464   \def\@gls@textaccess{#1}%
5465 }
5466 \define@key{glsxtrabrv}{firstaccess}{%
5467   \def\@gls@firstaccess{#1}%
5468 }
5469 \define@key{glsxtrabrv}{shortaccess}{%
5470   \def\@gls@shortaccess{#1}%
5471 }
5472 \define@key{glsxtrabrv}{shortpluralaccess}{%
5473   \def\@gls@shortaccesspl{#1}%
5474 }

@initaccesskeys
5475 \newcommand*{\@gls@initaccesskeys}{%
5476   \def\@gls@nameaccess{}%
5477   \def\@gls@textaccess{}%
5478   \def\@gls@firstaccess{}%
5479   \def\@gls@shortaccess{}%
5480   \def\@gls@shortaccesspl{}%
5481 }

```

essattribute@set

```
\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}
```

```
5482 \newcommand*{\@gls@ifaccessattribute@set}[3]{%
5483   \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
5484   {#2}%
5485   {%
5486     \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
5487     {#3}%
5488     {%
5489       \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
5490       {#2}%
5491       {#3}%
5492     }%
5493   }%
5494 }
```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation.

```
5495 \newcommand{\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```
5496 \ifdefempty\@gls@shortaccess
5497 {%
5498   \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
5499   {%
5500     \@glsxtr@insertdots\@gls@shortaccess{#1}%
5501     \eappto\ExtraCustomAbbreviationFields{%
5502       shortaccess={\expandonce\@gls@shortaccess},}%
5503     }%
5504   }%
5505 }%
5506 }
```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```
5507 \ifdefempty\@gls@shortaccess
5508 {}%
5509 {%
5510   \ifdefempty\@gls@shortaccesspl
5511   {%
5512     \@gls@ifaccessattribute@set{aposplural}%
5513     {%
5514       \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5515         \@gls@shortaccess'\abbrvpluralsuffix}%
5516     }%
5517   }%
5518   \@gls@ifaccessattribute@set{noshortplural}%
5519   {%
```

```

5520         \let\@gls@shortaccesspl\@gls@shortaccess
5521     }%
5522     {%
5523         \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
5524             \@gls@shortaccess\abbrvpluralsuffix}%
5525     }%
5526 }%
5527 \eappto\ExtraCustomAbbreviationFields{%
5528     shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
5529 }%
5530 {}%
5531 }%

```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```

5532 \ifdefempty\@gls@nameaccess
5533 {%
5534     \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
5535     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5536         \ifdefempty\@gls@shortaccess
5537         {}%
5538         {%
5539             \eappto\ExtraCustomAbbreviationFields{%
5540                 access={\expandonce\@gls@shortaccess},%
5541             }%
5542         }%
5543     }%
5544     {}%
5545 }%
5546 {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

5547 \ifdefempty\@gls@textaccess
5548 {%
5549     \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
5550     {%

```

Do nothing if the shortaccess key hasn't been set.

```

5551         \ifdefempty\@gls@shortaccess
5552         {}%
5553         {%
5554             \eappto\ExtraCustomAbbreviationFields{%
5555                 textaccess={\expandonce\@gls@shortaccess},%
5556             }%
5557         }%
5558     }%
5559     {}%
5560 }%
5561 {}%

```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```
5562 \ifdefempty\@gls@firstaccess
5563 {%
5564 \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
5565 {%
```

Do nothing if the shortaccess key hasn't been set.

```
5566 \ifdefempty\@gls@shortaccess
5567 {}%
5568 {%
5569 \eappto\ExtraCustomAbbreviationFields{%
5570 firstaccess={\expandonce\@gls@shortaccess},%
5571 }%
5572 }%
5573 }%
5574 {}%
5575 }%
5576 {}%
5577 }
```

End of if accsupp part

```
5578 }
5579 {
```

No accessibility support. Just define these commands to do \glsentry<xxx>

```
\glsaccessname Display the name value (no link and no check for existence).
5580 \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.
5581 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.
5582 \newcommand*{\GLSaccessname}[1]{%
5583 \protect\mfirstucMakeUppercase{\glsentryname{#1}}}}

\glsaccesstext Display the text value (no link and no check for existence).
5584 \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.
5585 \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.
5586 \newcommand*{\GLSaccesstext}[1]{%
5587 \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}}

glsaccessplural Display the plural value (no link and no check for existence).
5588 \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.  
5589 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.  
5590 `\newcommand*{\GLSaccessplural}[1]{%`  
5591 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).  
5592 `\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.  
5593 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.  
5594 `\newcommand*{\GLSaccessfirst}[1]{%`  
5595 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).  
5596 `\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.  
5597 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.  
5598 `\newcommand*{\GLSaccessfirstplural}[1]{%`  
5599 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`glsaccesssymbol` Display the symbol value (no link and no check for existence).  
5600 `\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.  
5601 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.  
5602 `\newcommand*{\GLSaccesssymbol}[1]{%`  
5603 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence).  
5604 `\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.  
5605 `\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

esssymbolplural Display the symbolplural value (no link and no check for existence). converted to upper case.

```
5606 \newcommand*{\GLSaccesssymbolplural}[1]{%
5607 \protect\mfirstucMakeUppercase{\glentrysymbolplural{#1}}}
```

\glsaccessdesc Display the desc value (no link and no check for existence).

```
5608 \newcommand*{\glsaccessdesc}[1]{\glentrydesc{#1}}
```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```
5609 \newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}
```

\GLSaccessdesc Display the desc value (no link and no check for existence). converted to upper case.

```
5610 \newcommand*{\GLSaccessdesc}[1]{%
5611 \protect\mfirstucMakeUppercase{\glentrydesc{#1}}}
```

ccessdescplural Display the descplural value (no link and no check for existence).

```
5612 \newcommand*{\glsaccessdescplural}[1]{\glentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```
5613 \newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence). converted to upper case.

```
5614 \newcommand*{\GLSaccessdescplural}[1]{%
5615 \protect\mfirstucMakeUppercase{\glentrydescplural{#1}}}
```

\glsaccessshort Display the short form (no link and no check for existence).

```
5616 \newcommand*{\glsaccessshort}[1]{\glentryshort{#1}}
```

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```
5617 \newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}
```

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.

```
5618 \newcommand*{\GLSaccessshort}[1]{%
5619 \protect\mfirstucMakeUppercase{\glentryshort{#1}}}
```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```
5620 \newcommand*{\glsaccessshortpl}[1]{\glentryshortpl{#1}}
```

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```
5621 \newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

**LSaccessshortpl** Display the shortplural value (no link and no check for existence). converted to upper case.

```
5622 \newcommand*{\GLSaccessshortpl}[1]{%
5623 \protect\mfirstucMakeUppercase{\glentryshortpl{#1}}}
```

**\glsaccesslong** Display the long form (no link and no check for existence).

```
5624 \newcommand*{\glsaccesslong}[1]{\glentrylong{#1}}
```

**\Glsaccesslong** Display the long form (no link and no check for existence).

```
5625 \newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}
```

**\GLSaccesslong** Display the long value (no link and no check for existence). converted to upper case.

```
5626 \newcommand*{\GLSaccesslong}[1]{%
5627 \protect\mfirstucMakeUppercase{\glentrylong{#1}}}
```

**glsaccesslongpl** Display the long plural form (no link and no check for existence).

```
5628 \newcommand*{\glsaccesslongpl}[1]{\glentrylongpl{#1}}
```

**Glsaccesslongpl** Display the long plural form (no link and no check for existence).

```
5629 \newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

**GLSaccesslongpl** Display the longplural value (no link and no check for existence). converted to upper case.

```
5630 \newcommand*{\GLSaccesslongpl}[1]{%
5631 \protect\mfirstucMakeUppercase{\glentrylongpl{#1}}}
```

**@initaccesskeys** This does nothing if there's no accessibility support.

```
5632 \newcommand*{\@gls@initaccesskeys}{}
```

**lt@short@access** This does nothing if there's no accessibility support.

```
5633 \newcommand{\@gls@setup@default@short@access}[1]{}%
```

End of else part

```
5634 }
```

## 1.6 Categories

**\glscategory** Add a new storage key that can be used to indicate a category. The default category is general.

```
5635 \glsaddstoragekey{category}{general}{\glscategory}
```

**\glsifcategory** Convenient shortcut to determine if an entry has the given category.

```
5636 \newcommand{\glsifcategory}[4]{%
5637 \ifglsseldeq{#1}{category}{#2}{#3}{#4}%
5638 }
```

Categories can have attributes.

categoryattribute

```
\glissetcategoryattribute{<category>}{<attribute-label>}{<value>}
```

Set (or override if already set) an attribute for the given category.

```
5639 \newcommand*{\glissetcategoryattribute}[3]{%
5640   \csdef{@glsxtr@categoryattr@@#1@#2}{#3}%
5641 }
```

categoryattribute

```
\glsggetcategoryattribute{<category>}{<attribute-label>}
```

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
5642 \newcommand*{\glsggetcategoryattribute}[2]{%
5643   \csuse{@glsxtr@categoryattr@@#1@#2}%
5644 }
```

categoryattribute

```
\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}
```

Tests if the category has the given attribute set.

```
5645 \newcommand*{\glshascategoryattribute}[4]{%
5646   \ifcsvoid{@glsxtr@categoryattr@@#1@#2}{#4}{#3}%
5647 }
```

\glissetattribute

```
\glissetattribute{<entry label>}{<attribute-label>}{<value>}
```

Short cut where the category label is obtained from the entry information.

```
5648 \newcommand*{\glissetattribute}[3]{%
5649   \glissetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5650 }
```

\glsggetattribute

```
\glsggetattribute{<entry label>}{<attribute-label>}
```

Short cut where the category label is obtained from the entry information.

```
5651 \newcommand*{\glsggetattribute}[2]{%
5652   \glsggetcategoryattribute{\glscategory{#1}}{#2}%
5653 }
```



\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.

```
5654 \newcommand*{\glshasattribute}[4]{%
5655   \ifglentryexists{#1}%
5656   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5657   {#4}%
5658 }
```

categoryattribute

```
\glcifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true
part>}{<false part>}
```

True if category has the attribute with the given value.

```
5659 \newcommand{\glcifcategoryattribute}[5]{%
5660   \ifcsundef{@glxtr@categoryattr@#1@#2}%
5661   {#5}%
5662   {\ifcsstring{@glxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
5663 }
```

\glcifattribute

```
\glcifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{
<false part>}
```

Short cut to determine if the given entry has a category with the given attribute set.

```
5664 \newcommand{\glcifattribute}[5]{%
5665   \ifglentryexists{#1}%
5666   {\glcifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5667   {#5}%
5668 }
```

Set attributes for the default general category:

```
5669 \glsetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
5670 \glsetcategoryattribute{acronym}{regular}{true}
```

regularcategory

Convenient shortcut to create add the regular attribute.

```
5671 \newcommand*{\glsetregularcategory}[1]{%
5672   \glsetcategoryattribute{#1}{regular}{true}%
5673 }
```

fregularcategory

```
\glsifregularcategory{<category>}{<true part>}{<false part>}
```

Short cut to determine if a category has the regular attribute explicitly set to true.

```
5674 \newcommand{\glsifregularcategory}[3]{%  
5675   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%  
5676 }
```

tregularcategory

```
\glsifnotregularcategory{<category>}{<true part>}{<false part>}
```

Short cut to determine if a category has the regular attribute explicitly set to false.

```
5677 \newcommand{\glsifnotregularcategory}[3]{%  
5678   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%  
5679 }
```

\glsifregular

```
\glsifregular{<entry label>}{<true part>}{<false part>}
```

Short cut to determine if an entry has a regular attribute set to true.

```
5680 \newcommand{\glsifregular}[3]{%  
5681   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%  
5682 }
```

\glsifnotregular

```
\glsifnotregular{<entry label>}{<true part>}{<false part>}
```

Short cut to determine if an entry has a regular attribute set to false.

```
5683 \newcommand{\glsifnotregular}[3]{%  
5684   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%  
5685 }
```

oreachincategory

```
\glsforeachincategory[<glossary labels>]{<category-label>}  
{<glossary-cs>}{<label-cs>}{<body>}
```

Iterates through all entries in all the glossaries (or just those listed in *<glossary labels>*) and does *<body>* if the category matches *<category-label>*. The control sequences *<glossary-cs>* and *<label-cs>* may be used in *<body>* to access the glossary label and entry label for the current iteration.

```
5686 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
```

```

5687 \forallglossaries[#1]{#3}%
5688 {%
5689     \forallsentries[#3]{#4}%
5690     {%
5691         \glsifcategory{#4}{#2}{#5}{}%
5692     }%
5693 }%
5694 }

```

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.

```

5695 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5696     \forallglossaries[#1]{#4}%
5697     {%
5698         \forallsentries[#4]{#5}%
5699         {%
5700             \glsifattribute{#5}{#2}{#3}{#6}{}%
5701         }%
5702     }%
5703 }

```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to index and add `\glstrpostdescription`.

```

5704 \ifdef\newterm
5705 {%

```

\newterm

```

5706     \renewcommand*{\newterm}[2][ ]{%
5707         \newglossaryentry{#2}%
5708         {type=index,category=index,name={#2},%
5709          description={\glstrpostdescription\nopostdesc},#1}%
5710     }

```

Indexed terms are regular by default.

```

5711     \glsssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

5712     \newcommand*{\glstrpostdescindex}{}

5713 }
5714 {}

```

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.

```
5715 \ifdef\printsymbols
5716 {%
```

`glsxtrnewsymbol` Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```
5717 \newcommand*{\glsxtrnewsymbol}[3] [] {%
5718 \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5719 }
```

Symbols are regular by default.

```
5720 \glsssetcategoryattribute{symbol}{regular}{true}
```

`rpostdescsymbol`

```
5721 \newcommand*{\glsxtrpostdescsymbol}{}

5722 }
5723 {}
```

Similar for the numbers option.

```
5724 \ifdef\printnumbers
5725 {%
```

`glsxtrnewnumber`

```
5726 \ifdef\printnumbers
5727 \newcommand*{\glsxtrnewnumber}[3] [] {%
5728 \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5729 }
```

Numbers are regular by default.

```
5730 \glsssetcategoryattribute{number}{regular}{true}
```

`rpostdescnumber`

```
5731 \newcommand*{\glsxtrpostdescnumber}{}

5732 }
5733 {}
```

`sxtrsetcategory` Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
5734 \newcommand*{\glsxtrsetcategory}[2] {%
5735 \@for\@glsxtr@label:=#1\do
5736 {%
5737 \glsfieldxddef{\@glsxtr@label}{category}{#2}%
5738 }%
5739 }
```

`\categoryforall` Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```

5740 \newcommand*{\glxtrsetcategoryforall}[2]{%
5741   \forallglossaries[#1]{\@glxtr@type}{%
5742     \forallglsentries[\@glxtr@type]{\@glxtr@label}%
5743     {%
5744       \glsfieldxdef{\@glxtr@label}{category}{#2}%
5745     }%
5746   }%
5747 }

```

`\glxtrfieldtitlecase` `\glxtrfieldtitlecase{\langle label \rangle}{\langle field \rangle}`

Apply title casing to the contents of the given field.

```

5748 \newcommand*{\glxtrfieldtitlecase}[2]{%
5749   \expandafter\glxtrfieldtitlecasesecs\expandafter
5750   {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
5751 }

```

`\fieldtitlecasesecs` The command used by `\glxtrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```

5752 \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.

```

5753 \ifpackageloaded{glossaries-accsupp}
5754 {
5755   \renewcommand*{\glossentrydesc}[1]{%
5756     \glsdoifexistsorwarn{#1}%
5757     {%
5758       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossdescfont` attribute to determine the font applied.

```

5759   \glshasattribute{#1}{glossdescfont}%
5760   {%
5761     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
5762     \ifcsdef{\@glxtr@attrval}%
5763     {%
5764       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5765     }%
5766     {%
5767       \GlossariesExtraWarning{Unknown control sequence name
5768         '\@glxtr@attrval' supplied in glossdescfont attribute

```

```

5769         for entry '#1'. Ignoring}%
5770         \let\@glxtr@glossdescfont\@firstofone
5771     }%
5772 }%
5773 {\let\@glxtr@glossdescfont\@firstofone}%
5774 \glsifattribute{#1}{glossdesc}{firstuc}%
5775 {%
5776     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5777 }%
5778 {%
5779     \glsifattribute{#1}{glossdesc}{title}%
5780 {%
5781     \@glxtr@do@titlecaps@warn
5782     \glsdescriptionaccessdisplay
5783     {%
5784         \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
5785     }%
5786     {#1}%
5787 }%
5788 {%
5789     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5790 }%
5791 }%
5792 }%
5793 }
5794 }
5795 {
5796 \renewcommand*{\glossentrydesc}[1]{%
5797     \glsdoifexistsorwarn{#1}%
5798     {%
5799         \glssetabbrvfmt{\glscategory{#1}}%
5800         \glsattribute{#1}{glossdescfont}%
5801         {%
5802             \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
5803             \ifcsdef{\@glxtr@attrval}%
5804             {%
5805                 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5806             }%
5807             {%
5808                 \GlossariesExtraWarning{Unknown control sequence name
5809                     '\@glxtr@attrval' supplied in glossdescfont attribute
5810                     for entry '#1'. Ignoring}%
5811                 \let\@glxtr@glossdescfont\@firstofone
5812             }%
5813         }%
5814         {\let\@glxtr@glossdescfont\@firstofone}%
5815         \glsifattribute{#1}{glossdesc}{firstuc}%
5816         {%
5817             \@glxtr@glossdescfont{\Glsentrydesc{#1}}%

```

```

5818 }%
5819 {%
5820   \glsifattribute{#1}{glossdesc}{title}%
5821   {%
5822     \@glsxtr@do@titlecaps@warn
5823     \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
5824   }%
5825   {%
5826     \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
5827   }%
5828 }%
5829 }%
5830 }
5831 }

```

`\glossentryname` If the `glossname` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

5832 \ifpackageloaded{glossaries-accsupp}
5833 {
5834   \renewcommand*{\glossentryname}[1]{%
5835     \@glsdoifexistsorwarn{#1}%
5836     {%
5837       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossnamefont` attribute to determine the font applied.

```

5838   \glsifattribute{#1}{glossnamefont}%
5839   {%
5840     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
5841     \ifcsdef{\@glsxtr@attrval}%
5842     {%
5843       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
5844     }%
5845     {%
5846       \GlossariesExtraWarning{Unknown control sequence name
5847         ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
5848         for entry ‘#1’. Reverting to default \string\glsnamefont}%
5849       \let\@glsxtr@glossnamefont\glsnamefont
5850     }%
5851   }%
5852   {\let\@glsxtr@glossnamefont\glsnamefont}%
5853   \glsifattribute{#1}{glossname}{firstuc}%
5854   {%
5855     \glsnameaccessdisplay
5856     {%
5857       \@glsxtr@glossnamefont{\Glsentryname{#1}}%
5858     }%
5859     {#1}%
5860   }%
5861   {%
5862     \glsifattribute{#1}{glossname}{title}%

```

```

5863      {%
5864      \@glstr@do@titlecaps@warn
5865      \glsnameaccessdisplay
5866      {%
5867      \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}%
5868      }%
5869      {#1}%
5870      }%
5871      {%
5872      \glsifattribute{#1}{glossname}{uc}%
5873      {%
5874      \glsnameaccessdisplay
5875      {%

```

Hide the label from the upper-casing command.

```

5876      \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
5877      \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5878      }%
5879      {#1}%
5880      }%
5881      {%
5882      \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
5883      \glsnameaccessdisplay
5884      {%
5885      \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
5886      }%
5887      {#1}%
5888      }%
5889      }%
5890      }%

```

Do post-name hook:

```

5891      \glstrpostnamehook{#1}%
5892      }%
5893    }
5894  }
5895  {
5896    \renewcommand*{\glossentryname}[1]{%
5897      \@glsdofexistsorwarn{#1}%
5898      {%
5899      \glsetabbrvfmt{\glscategory{#1}}%
5900      \glshasattribute{#1}{glossnamefont}%
5901      {%
5902      \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
5903      \ifcsdef{\@glstr@attrval}%
5904      {%
5905      \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
5906      }%
5907      {%
5908      \GlossariesExtraWarning{Unknown control sequence name

```



```

5909         ‘\@glsxtr@attrval’ supplied in glossnamefont attribute
5910         for entry ‘#1’. Reverting to default \string\glsnamefont}%
5911         \let\@glsxtr@glossnamefont\glsnamefont
5912     }%
5913 }%
5914 {\let\@glsxtr@glossnamefont\glsnamefont}%
5915 \glsifattribute{#1}{glossname}{firstuc}%
5916 {%
5917     \@glsxtr@glossnamefont{\Glsentryname{#1}}%
5918 }%
5919 {%
5920     \glsifattribute{#1}{glossname}{title}%
5921     {%
5922         \@glsxtr@do@titlecaps@warn
5923         \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
5924     }%
5925     {%
5926         \glsifattribute{#1}{glossname}{uc}%
5927         {%

```

Hide the label from the upper-casing command.

```

5928         \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
5929         \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5930     }%
5931     {%

```

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.

```

5932         \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
5933         \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
5934     }%
5935 }%
5936 }%

```

Do post-name hook.

```

5937     \glsxtrpostnamehook{#1}%
5938 }%
5939 }
5940 }

```

**\Glossentryname** Redefine to set the abbreviation format and accessibility support.

```

5941 \@ifpackageloaded{glossaries-accsupp}
5942 {
5943     \renewcommand*{\Glossentryname}[1]{%
5944         \@glsdoifexistsorwarn{#1}%
5945         {%
5946             \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

5947         \glschasattribute{#1}{glossnamefont}%
5948         {%

```

```

5949     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
5950     \ifcsdef{\@glxtr@attrval}%
5951     {%
5952         \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5953     }%
5954     {%
5955         \GlossariesExtraWarning{Unknown control sequence name
5956             '\@glxtr@attrval' supplied in glossnamefont attribute
5957             for entry '#1'. Reverting to default \string\glsnamefont}%
5958         \let\@glxtr@glossnamefont\glsnamefont
5959     }%
5960 }%
5961 {\let\@glxtr@glossnamefont\glsnamefont}%
5962 \glsnameaccessdisplay
5963 {%
5964     \@glxtr@glossnamefont{\Glsentryname{#1}}%
5965 }%
5966 {#1}%

```

Do post-name hook:

```

5967     \glxtrpostnamehook{#1}%
5968 }%
5969 }
5970 }
5971 {
5972     \renewcommand*{\Glossentryname}[1]{%
5973         \@glsdofexistsorwarn{#1}%
5974     }%
5975     \glsssetabbrvfmt{\glscategory{#1}}%
5976     \glshasattribute{#1}{glossnamefont}%
5977     {%
5978         \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
5979         \ifcsdef{\@glxtr@attrval}%
5980         {%
5981             \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5982         }%
5983         {%
5984             \GlossariesExtraWarning{Unknown control sequence name
5985                 '\@glxtr@attrval' supplied in glossnamefont attribute
5986                 for entry '#1'. Reverting to default \string\glsnamefont}%
5987             \let\@glxtr@glossnamefont\glsnamefont
5988         }%
5989     }%
5990     {\let\@glxtr@glossnamefont\glsnamefont}%
5991     \@glxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

5992     \glxtrpostnamehook{#1}%
5993 }%
5994 }

```

5995 }

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`\glstrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```
5996 \newcommand*{\glstrpostnamehook}[1]{%
5997   \let\@glsnumberformat\@glstr@defaultnumberformat
5998   \glstrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
5999   \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
6000   \csuse{glstrpostname\glscategory{#1}}%
6001 }
```

`\glsextrapostnamehook`

```
6002 \newcommand*{\glsextrapostnamehook}[1]{}%
```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```
6003 \newcommand*{\glsdefpostname}[2]{%
6004   \csdef{glstrpostname#1}{#2}%
6005 }
```

`\glssetaccessdisplay`

```
6006 \@ifpackageloaded{glossaries-accsupp}
6007 {
6008   \newcommand*{\glstrsetaccessdisplay}[1]{%
6009     \ifcsdef{gls#1accessdisplay}%
6010       {\letcs\@glstr@accessdisplay{gls#1accessdisplay}}%
6011       {%
```

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.

```
6012     \edef\@gls@thisval{#1}%
6013     \@for\@gls@map:=\@gls@keymap\do{%
6014       \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
6015       \ifdefequal{\@this@key}{\@gls@thisval}%
6016       {%
6017         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
6018         \@endfortrue
6019       }%
6020     }%
6021   }%
6022   \ifcsdef{gls\@gls@thisval accessdisplay}%
6023   {\letcs\@glstr@accessdisplay{gls\@gls@thisval accessdisplay}}%
```

```

6024      {\let\@glxtr@accessdisplay\@firstoftwo}%
6025    }%
6026  }
6027 }
6028 {%
6029   \newcommand*{\glxtr@setaccessdisplay}[1]{%
6030     \let\@glxtr@accessdisplay\@firstoftwo}
6031 }

```

sentrynameother Provide a command that works like \glossentryname but accesses a different field (which must be supplied using its internal field label).

```

6032 \newrobustcmd*{\glossentrynameother}[2]{%
6033   \@glsdofexistsorwarn{#1}%
6034   {%

```

Accessibility support:

```

6035   \glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6036   \glsssetabbrvfmt{\glscategory{#1}}%
6037   \glshasattribute{#1}{glossnamefont}%
6038   {%
6039     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
6040     \ifcsdef{\@glxtr@attrval}%
6041     {%
6042       \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6043     }%
6044     {%
6045       \GlossariesExtraWarning{Unknown control sequence name
6046         '\@glxtr@attrval' supplied in glossnamefont attribute
6047         for entry '#1'. Reverting to default \string\glssnamefont}%
6048       \let\@glxtr@glossnamefont\glssnamefont
6049     }%
6050   }%
6051   {\let\@glxtr@glossnamefont\glssnamefont}%
6052   \glssifattribute{#1}{glossname}{firstuc}%
6053   {%
6054     \@glxtr@accessdisplay
6055     {\@glxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6056     {#1}%
6057   }%
6058   {%
6059     \glssifattribute{#1}{glossname}{title}%
6060     {%
6061       \@glxtr@do@titlecaps@warn
6062       \@glxtr@accessdisplay
6063       {\@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{#2}}}%
6064       {#1}%
6065     }%
6066     {%

```

```

6067      \glsifattribute{#1}{glossname}{uc}%
6068      {%
6069      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6070      \@glsxtr@accessdisplay
6071      {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6072      {#1}%
6073      }%
6074      {%
6075      \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6076      \@glsxtr@accessdisplay
6077      {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6078      {#1}%
6079      }%
6080      }%
6081      }%

```

Do post-name hook.

```

6082      \glsxtrpostnamehook{#1}%
6083      }%
6084  }

```

`format@override` Determines if the format key should override the indexing attribute value.

```

6085 \newif\if@glsxtr@format@override
6086 \@glsxtr@format@overridefalse

```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

6087 \@ifpackageloaded{hyperref}
6088 {
  If hyperref's hyperindex option is on, then hyperref will automatically add \hyperpage, so
  don't add it.
6089   \ifHy@hyperindex
6090     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6091       \@glsxtr@format@override true
6092       \appto\theindex{\let\glshypernumber\@firstofone}%
6093     }
6094   \else
6095     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6096       \@glsxtr@format@override true
6097       \appto\theindex{\let\glshypernumber\hyperpage}%
6098     }
6099   \fi
6100 }
6101 {
6102   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6103     \@glsxtr@format@override true
6104   }

```

```

6105 }
6106 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

6107 \newcommand*{\glstrdoautoindexname}[2]{%
6108   \glshasattribute{#1}{#2}%
6109   {%

```

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.

```

6110   \@glstr@autoindex@setname{#1}%

```

If the attribute value is simply “true” don't add an encap, otherwise use the value as the encap.

```

6111   \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
6112   \if@glstr@format@override

6113     \ifx\@glstrnumberformat\@glstr@defaultnumberformat
6114     \else
6115       \let\@glstr@attrval\@glstrnumberformat
6116     \fi
6117   \fi
6118   \ifdefstring{\@glstr@attrval}{true}%
6119   {}%
6120   {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
6121   \expandafter\glstrautoindex\expandafter{\@glo@name}%
6122   }%
6123   {}%
6124 }

```

glstrautoindex

```

6125 \newcommand*{\glstrautoindex}{\index}

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

6126 \newcommand*{\@glstr@autoindex@setname}[1]{%
6127   \protected@edef\@glo@name{\glstrautoindexentry{#1}}%
6128   \glstrautoindexassignsort{\@glo@sort}{#1}%
6129   \@gls@checkmkidxchars\@glo@sort
6130   \@glstr@autoindex@doextra@esc\@glo@sort
6131   \epreto\@glo@name{\@glo@sort\@glstr@autoindex@at}%
6132 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

6133 \newcommand*{\glstrautoindexentry}[1]{\string\glstryname{#1}}

```

indexassignsort Used to assign the sort value when auto-indexing.

```

6134 \newcommand*{\glstrautoindexassignsort}[2]{%
6135   \glslentryfield{#1}{#2}{sort}%
6136 }

```

dex@doextra@esc

```
6137 \newcommand*{\@glsxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
6138 \ifx\@glsxtr@autoindex@esc\@gls@quotechar
6139 \else
6140 \def\@gls@checkedmkidx{}%
6141 \edef\@glsxtr@checkspch{%
6142 \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
6143 \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
6144 \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6145 @@glsxtr@checkspch
6146 \let#1\@gls@checkedmkidx\relax
6147 \fi
```

Escape actual character unless it has already been escaped.

```
6148 \ifx\@glsxtr@autoindex@at\@gls@actualchar
6149 \else
6150 \def\@gls@checkedmkidx{}%
6151 \edef\@glsxtr@checkspch{%
6152 \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
6153 \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
6154 \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6155 @@glsxtr@checkspch
6156 \let#1\@gls@checkedmkidx\relax
6157 \fi
```

Escape level character unless it has already been escaped.

```
6158 \ifx\@glsxtr@autoindex@level\@gls@levelchar
6159 \else
6160 \def\@gls@checkedmkidx{}%
6161 \edef\@glsxtr@checkspch{%
6162 \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
6163 \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
6164 \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6165 @@glsxtr@checkspch
6166 \let#1\@gls@checkedmkidx\relax
6167 \fi
```

Escape encap character unless it has already been escaped.

```
6168 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
6169 \else
6170 \def\@gls@checkedmkidx{}%
6171 \edef\@glsxtr@checkspch{%
6172 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
6173 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
6174 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6175 @@glsxtr@checkspch
6176 \let#1\@gls@checkedmkidx\relax
6177 \fi
6178 }
```

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`.

```
6179 \newcommand*{\@glxtr@autoindex@at}{}

```

`\trSetActualChar` Set the actual character.

```
6180 \newcommand*{\GlsXtrSetActualChar}[1]{%
6181   \gdef\@glxtr@autoindex@at{#1}%
6182   \def\@glxtr@autoindex@escat##1##2##3\@glxtr@endescspch{%
6183     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escat}{##1}{##2}{##3}%
6184   }%
6185 }
6186 \@onlypreamble\GlsXtrSetActualChar
6187 \makeatother
6188 \GlsXtrSetActualChar{@}
6189 \makeatletter

```

`\autoindex@encap` Encap character for use with `\index`.

```
6190 \newcommand*{\@glxtr@autoindex@encap}{}

```

`\XtrSetEncapChar` Set the encap character.

```
6191 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6192   \gdef\@glxtr@autoindex@encap{#1}%
6193   \def\@glxtr@autoindex@escencap##1##2##3\@glxtr@endescspch{%
6194     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escencap}{##1}{##2}{##3}%
6195   }%
6196 }
6197 \GlsXtrSetEncapChar{||}
6198 \@onlypreamble\GlsXtrSetEncapChar

```

`\autoindex@level` Level character for use with `\index`.

```
6199 \newcommand*{\@glxtr@autoindex@level}{}

```

`\XtrSetLevelChar` Set the encap character.

```
6200 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6201   \gdef\@glxtr@autoindex@level{#1}%
6202   \def\@glxtr@autoindex@esclevel##1##2##3\@glxtr@endescspch{%
6203     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@esclevel}{##1}{##2}{##3}%
6204   }%
6205 }
6206 \GlsXtrSetLevelChar{!}
6207 \@onlypreamble\GlsXtrSetLevelChar

```

`\r@autoindex@esc` Escape character for use with `\index`.

```
6208 \newcommand*{\@glxtr@autoindex@esc}{"}

```



`\lsXtrSetEscChar` Set the escape character.

```
6209 \newcommand*{\GlsXtrSetEscChar}[1]{%
6210   \gdef\@glsxtr@autoindex@esc{#1}%
6211   \def\@glsxtr@autoindex@escquote##1##2##3\@glsxtr@endescspch{%
6212     \@glsxtr@autoindex@escspch{#1}\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
6213   }%
6214 }
6215 \GlsXtrSetEscChar{"}
6216 \@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character `\actualchar`:

```
6217 \ifdef\actualchar
6218   {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6219 }
```

Quote character `\quotechar`:

```
6220 \ifdef\quotechar
6221   {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6222 }
```

Level character `\levelchar`:

```
6223 \ifdef\levelchar
6224   {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6225 }
```

Encap character `\encapchar`:

```
6226 \ifdef\encapchar
6227   {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6228 }
```

`\leto@endescspch`

```
6229 \def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

`\toindex@esc@spch`

<code>\@glsxtr@autoindex@escspch{&lt;char&gt;}{&lt;cs&gt;}{&lt;pre&gt;}{&lt;mid&gt;}{&lt;post&gt;}</code>
---

```
6230 \newcommand*{\@glsxtr@autoindex@escspch}[5]{%
6231   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
6232   \toks@={#3}%
6233   \ifx\@nnil#3\relax
6234     \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
6235   \else
6236     \ifx\@nnil#4\relax
6237       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
6238       \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
6239         #4#5\@glsxtr@endescspch}%
6240     \else
6241       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
```

```

6242      \@glxtr@autoindex@esc#1}%
6243      \def\@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
6244      \fi
6245      \fi
6246      \@glxtr@checkspch
6247 }

```

**\Glossentrydesc** Redefine to set the abbreviation format and accessibility support.

```

6248 \renewcommand*{\Glossentrydesc}[1]{%
6249   \glsoifexistsorwarn{#1}%
6250   {%
6251     \glsetabbrvfmt{\glscategory{#1}}%
6252     \Glsaccessdesc{#1}%
6253   }%
6254 }

```

**\Glossentrysymbol** Redefine to set the abbreviation format and accessibility support.

```

6255 \renewcommand*{\Glossentrysymbol}[1]{%
6256   \glsoifexistsorwarn{#1}%
6257   {%
6258     \glsetabbrvfmt{\glscategory{#1}}%
6259     \Glsaccesssymbol{#1}%
6260   }%
6261 }

```

**\Glossentrysymbol** Redefine to set the abbreviation format and accessibility support.

```

6262 \renewcommand*{\Glossentrysymbol}[1]{%
6263   \glsoifexistsorwarn{#1}%
6264   {%
6265     \glsetabbrvfmt{\glscategory{#1}}%
6266     \Glsaccesssymbol{#1}%
6267   }%
6268 }

```

Allow initials to be marked but only use the formatting for the tag in the glossary.

**\GlsXtrEnableInitialTagging** Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```

6269 \newcommand*{\GlsXtrEnableInitialTagging}{%
6270   \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
6271 }
6272 \@onlypreamble\GlsXtrEnableInitialTagging

```

**\GlsXtrEnableInitialTagging** Starred version undefines command.

```

6273 \newcommand*{\s@glxtr@enabletagging}[2]{%
6274   \undef#2%
6275   \@glxtr@enabletagging{#1}{#2}%
6276 }

```

r@enabletagging Internal command.

```
6277 \newcommand*{\@glsxtr@enabletagging}[2]{%
```

Set attributes for categories given in the first argument.

```
6278 \for\@glsxtr@cat:=#1\do
6279 {%
6280 \ifdefempty\@glsxtr@cat
6281 {}%
6282 {\glssetcategoryattribute{\@glsxtr@cat}{tagging}{true}}%
6283 }%
6284 \newrobustcmd*#2[1]{##1}%
6285 \def\@glsxtr@taggingcs{#2}%
6286 \renewcommand*\@glsxtr@activate@initialtagging{%
6287 \let#2\@glsxtr@tag
6288 }%
6289 \ifundef\@gls@preglossaryhook
6290 {\GlossariesExtraWarning{Initial tagging requires at least
6291 glossaries.sty v4.19 to work correctly}}%
6292 {}%
6293 }
```

Are we using an old version of mfirstuc that has a bug in \capitalisewords? If so, patch it so we don't have a problem with a combination of tagging and title case.

fu@checkword@do If this command hasn't been defined, then we have pre v2.02 of mfirstuc

```
6294 \ifundef\mfu@checkword@do
6295 {
6296 \newcommand*{\mfu@checkword@do}[1]{%
6297 \ifdefstring{\mfu@checkword@arg}{#1}%
6298 {%
6299 \let\@mfu@domakefirstuc\@firstofone
6300 \listbreak
6301 }%
6302 {}%
6303 }
```

\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.

```
6304 \ifundef\mfu@checkword
6305 {
6306 \newcommand{\@glsxtr@do@titlecaps@warn}{%
6307 \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
6308 support not available}}%
```

One warning should suffice.

```
6309 \let\@glsxtr@do@titlecaps@warn\relax
6310 }
6311 }
6312 {
6313 \renewcommand*{\mfu@checkword}[1]{%
```

```

6314      \def\mfu@checkword@arg{#1}%
6315      \let\@mfu@domakefirstuc\makefirstuc
6316      \forlistloop\mfu@checkword@do\@mfu@nocaplist
6317    }
6318  }
6319}
6320{}% no patch required

@titlecaps@warn  Do warning if title case not supported.
6321 \newcommand*{\@glstr@do@titlecaps@warn}{}

@initialtagging  Used in \printglossary but at least v4.19 of glossaries required.
6322 \newcommand*{\@glstr@activate@initialtagging}{}

\@glstr@tag  Definition of tagging command when used in glossary.
6323 \newrobustcmd*{\@glstr@tag}[1]{%
6324   \gl@ifattribute{\glscurrententrylabel}{tagging}{true}%
6325   {\glstrtagfont{#1}}{#1}%
6326 }

\glstrtagfont  Used in the glossary.
6327 \newcommand*{\glstrtagfont}[1]{\underline{#1}}

preglossaryhook  This macro was introduced in glossaries version 4.19, so it may not be defined. If it hasn't
                  been defined this feature is unavailable. A check is added for the entry's existence to prevent
                  errors from occurring if the user removes an entry or changes the label, which can interrupt
                  the build process.
6328 \ifdef\@glstr@preglossaryhook
6329 {
6330   \renewcommand*{\@glstr@preglossaryhook}{%
6331     \@glstr@activate@initialtagging
        Since the glossaries are automatically scoped, \@glstr@org@postdescription shouldn't
        already be defined, but check anyway just as a precautionary measure.
6332     \ifundef\@glstr@org@postdescription
6333     {%
6334       \let\@glstr@org@postdescription\glspostdescription
6335       \renewcommand*{\glspostdescription}{%
6336         \ifglentryexists{\glscurrententrylabel}%
6337         {%
6338           \glstrpostdescription
6339           \@glstr@org@postdescription
6340         }%
6341         {}%
6342       }%
6343     }%
6344     {}%

```

Enable the options used by \@glsxtrp:

```
6345 \glossxtrsetpopts
6346 }%
6347 }
6348 }
```

**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.

```
6349 \newcommand*{\glsxtrpostdescription}{%
6350 \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
6351 }
```

**postdescgeneral**

```
6352 \newcommand*{\glsxtrpostdescgeneral}{}%
```

**xtrpostdescterm**

```
6353 \newcommand*{\glsxtrpostdescterm}{}%
```

**postdescacronym**

```
6354 \newcommand*{\glsxtrpostdescacronym}{}%
```

**descabbreviation**

```
6355 \newcommand*{\glsxtrpostdescabbreviation}{}%
```

**\glsdefpostdesc** Provide a convenient command for defining the post-description hook for the given category.

```
6356 \newcommand*{\glsdefpostdesc}[2]{%
6357 \csdef{glsxtrpostdesc#1}{#2}%
6358 }
```

**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.

```
6359 \renewcommand*{\glspostlinkhook}{%
6360 \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
6361 }
```

**xtrpostlinkhook** The entry label should already be stored in \glslabel by \@gls@link.

```
6362 \newcommand*{\glsxtrpostlinkhook}{%
6363 \glsxtrdiscardperiod{\glslabel}%
6364 {\glsxtrpostlinkendsentence}%
6365 {\glsxtrifcustomdiscardperiod
6366 {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}}%
6367 {\glsxtrpostlink}%
6368 }%
6369 }
```

`omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
6370 \newcommand*{\glxtrifcustomdiscardperiod}[2]{#2}
```

`\glxtrpostlink`

```
6371 \newcommand*{\glxtrpostlink}{%
6372   \csuse{\glxtrpostlink\glscategory{\glslabel}}%
6373 }
```

`\glsdefpostlink` Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite `\glxtrpostlink`.

```
6374 \newcommand*{\glsdefpostlink}[2]{%
    \ifthenelse{\equal{#1}{}}{%
6375       \PackageError{glossaries-extra}
6376       {Invalid empty category label in \string\glsdefpostlink}{}}%
6377       {\csdef{\glxtrpostlink#1}{#2}}%
6378       {\csdef{\glxtrpostlink#1}{#2}}%
6379 }
```

`linkendsentence` Done by `\glxtrpostlinkhook` if a full stop is discarded.

```
6380 \newcommand*{\glxtrpostlinkendsentence}{%
6381   \ifcsdef{\glxtrpostlink\glscategory{\glslabel}}
6382   {%
6383     \csuse{\glxtrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
6384     .\spacefactor\sfcode'\. \relax
6385   }%
6386   {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the spacefactor.

```
6387     \spacefactor\sfcode'\. \relax
6388   }%
6389 }
```

`dDescOnFirstUse` Provide a command for appending the description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
6390 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
6391   \glxtrifwasfirstuse{\space\glxtrparen{\glaccessdesc{\glslabel}}}{}%
6392 }
```

`ymbolOnFirstUse` Provide a command for appending the symbol (if defined) in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
6393 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
6394   \glxtrifwasfirstuse
6395   {%
```

```

6396     \ifglshassymbol{\glslabel}%
6397     {\space\glsxtrparen{\glssaccesssymbol{\glslabel}}}%
6398     }%
6399 }%
6400 {}%
6401 }

```

**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.

```

6402 \newcommand*{\glsxtrpostlinkAddSymbolDescOnFirstUse}{%
6403   \glsxtrifwasfirstuse
6404   {%
6405     \space\glsxtrparen
6406     {%
6407       \ifglshassymbol{\glslabel}%
6408       {\glssaccesssymbol{\glslabel}, }%
6409       }%
6410       \glssaccessdesc{\glslabel}%
6411     }%
6412   }%
6413   {}%
6414 }

```

**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).

```

6415 \newcommand*{\glsxtrdiscardperiod}[3]{%
6416   \glsxtrifwasfirstuse
6417   {%
6418     \glssifattribute{#1}{retainfirstuseperiod}{true}%
6419     {#3}%
6420     {%
6421       \glssifattribute{#1}{discardperiod}{true}%
6422       {%
6423         \glssifplural
6424         {%
6425           \glssifattribute{#1}{pluraldiscardperiod}{true}%
6426           {\glsxtrifperiod{#2}{#3}}%
6427           {#3}%
6428         }%
6429         {%
6430           \glsxtrifperiod{#2}{#3}%
6431           }%
6432         }%
6433       {#3}%
6434     }%
6435   }%
6436   {%

```

```

6437 \glsifattribute{#1}{discardperiod}{true}%
6438 {%
6439 \glsifplural
6440 {%
6441 \glsifattribute{#1}{pluraldiscardperiod}{true}%
6442 {\glsxtrifperiod{#2}{#3}}%
6443 {#3}%
6444 }%
6445 {%
6446 \glsxtrifperiod{#2}{#3}%
6447 }%
6448 }%
6449 {#3}%
6450 }%
6451 }

```

`\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`

```

6452 \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 ' ').

```

6453 \newcommand*{\glsxtr@punclist}{.,:;?!}

```

`\punctuationmark` Add character to punctuation list.

```

6454 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}

```

`\punctuationmarks` Reset the punctuation list.

```

6455 \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`.)

```

6456 \newcommand*{\glsxtrifnextpunc}[2]{%
6457 \def\reserved@a{#1}%
6458 \def\reserved@b{#2}%
6459 \futurelet\@glspunc@token\glsxtr@ifnextpunc
6460 }

```

`\glsxtr@ifnextpunc`

```

6461 \newcommand*{\glsxtr@ifnextpunc}{%
6462 \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{-}%
6463 \reserved@b
6464 }

```



```

xtr@ifpunctoken  Test if the token given in the first argument is in the punctuation list.
6465 \newcommand*{\glxtr@ifpunctoken}[1]{%
6466   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
6467 }

```

```

xtr@ifpunctoken
6468 \def\@glxtr@ifpunctoken#1#2{%
6469   \let\reserved@d=#2%
6470   \ifx\reserved@d\@nnil
6471     \let\glxtr@next\@glxtr@notfoundinlist
6472   \else
6473     \ifx#1\reserved@d
6474       \let\glxtr@next\@glxtr@foundinlist
6475     \else
6476       \let\glxtr@next\@glxtr@ifpunctoken
6477     \fi
6478   \fi
6479   \glxtr@next#1%
6480 }

```

```

xtr@foundinlist
6481 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}

```

```

@notfoundinlist
6482 \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.

```

6483 \newcommand{\glxtrdopostpunc}[1]{%
6484   \glxtr@ifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
6485 }

```

```

@glxtr@swaptwo
6486 \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.

```

6487 \define@key{glsxtrabbrv}{category}{%
6488   \edef\glscategorylabel{#1}%
6489   \ifcsdef{@glsabbrv@current@#1}%
6490   {%

```

Warning should already have been issued.

```

6491   \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6492   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6493   \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
6494   \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
6495 }%
6496 {}%
6497 }

```

Save the short plural form. This may be needed before the entry is defined.

```

6498 \define@key{glsxtrabbrv}{shortplural}{%
6499   \def\@gls@shortpl{#1}%
6500 }

```

Similarly for the long plural form.

```

6501 \define@key{glsxtrabbrv}{longplural}{%
6502   \def\@gls@longpl{#1}%
6503 }

```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```

6504 \newtoks\glsshortpltok

```

\glslongpltok

```

6505 \newtoks\glslongpltok

```

**glsxtr@insertdots** Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to \newabbreviation. Note that explicitly using the short or shortplural keys will override this.

```

6506 \newcommand*{\@glsxtr@insertdots}[2]{%
6507   \def#1{%
6508     \@glsxtr@insert@dots#1#2\@nnil
6509 }

```

**glsxtr@insert@dots**

```

6510 \newcommand*{\@glsxtr@insert@dots}[2]{%
6511   \ifx\@nnil#2\relax
6512   \let\@glsxtr@insert@dots@next\@gobble
6513   \else
6514   \ifx\relax#2\relax

```

```

6515 \else
6516 \appto#1{#2.}%
6517 \fi
6518 \let\@glxstr@insert@dots@next\@glxstr@insert@dots
6519 \fi
6520 \@glxstr@insert@dots@next#1%
6521 }

```

Similarly provide a way of replacing spaces with \glxstrwordsep, which first needs to be defined:

\glxstrwordsep

```

6522 \newcommand*{\glxstrwordsep}{\space}

```

Each word is marked with

\glxstrword

```

6523 \newcommand*{\glxstrword}[1]{#1}

```

tr@markwordseps

```

6524 \newcommand*{\@glxstr@markwordseps}[2]{%
6525 \def#1{}%
6526 \@glxstr@mark@wordseps#1#2 \@nnil
6527 }

```

r@mark@wordseps

```

6528 \def\@glxstr@mark@wordseps#1#2 #3{%
6529 \ifdefempty{#1}%
6530 {\def#1{\protect\glxstrword{#2}}}%
6531 {\appto#1{\protect\glxstrwordsep\protect\glxstrword{#2}}}%
6532 \ifx\@nnil#3\relax
6533 \let\@glxstr@mark@wordseps@next\relax
6534 \else
6535 \def\@glxstr@mark@wordseps@next{%
6536 \@glxstr@mark@wordseps#1#3}%
6537 \fi
6538 \@glxstr@mark@wordseps@next
6539 }

```

newabbreviation Define a new generic abbreviation.

```

6540 \newcommand*{\newabbreviation}[4][{}]{%
6541 \glxstr@newabbreviation{#1}{#2}{#3}{#4}%
6542 }

```

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.)

```

6543 \newcommand*{\glxstr@newabbreviation}[4]{%
6544 \gliskeylisttok{#1}%
6545 \glslabeltok{#2}%

```

6546 \glsshorttok{#3}%

6547 \glslongtok{#4}%

Save the original short and long values (before attribute settings modify them).

6548 \def\glstrorgshort{#3}%

6549 \def\glstrorglong{#4}%

Provide extra settings for hooks (if modified, this command must end with a comma).

6550 \def\ExtraCustomAbbreviationFields{}%

Initialise accessibility settings if required.

6551 \@gls@initaccesskeys

Get the category.

6552 \def\glscategorylabel{abbreviation}%

6553 \glstr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%

Ignore the shortplural and longplural keys.

6554 \setkeys\*{glstrabbrv}[shortplural,longplural]{#1}%

Set the default long plural

6555 \def\@gls@longpl{#4\glspluralsuffix}%

6556 \let\@gls@default@longpl\@gls@longpl

Has the markwords attribute been set?

6557 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%

6558 {%

6559 \@glstr@markwordseps\@gls@long{#4}%

6560 \expandafter\def\expandafter\@gls@longpl\expandafter

6561 {\@gls@long\glspluralsuffix}%

6562 \let\@gls@default@longpl\@gls@longpl

Update \glslongtok.

6563 \expandafter\glslongtok\expandafter{\@gls@long}%

6564 }%

6565 {}%

Has the markshortwords attribute been set? (Not compatible with insertdots.)

6566 \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%

6567 {%

6568 \@glstr@markwordseps\@gls@short{#3}%

6569 }%

6570 {}%

Has the insertdots attribute been set?

6571 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%

6572 {%

6573 \@glstr@insertdots\@gls@short{#3}%

6574 \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%

6575 }%

6576 {\def\@gls@short{#3}}%

6577 }%

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```
6578 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%  
6579 {%  
6580 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short  
6581 '\abbrvpluralsuffix}%  
6582 }%  
6583 {%
```

Has the noshortplural attribute been set?

```
6584 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%  
6585 {%  
6586 \let\@gls@shortpl\@gls@short  
6587 }%  
6588 {%  
6589 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short  
6590 \abbrvpluralsuffix}%  
6591 }%  
6592 }%
```

Update \glsshorttok:

```
6593 \expandafter\glsshorttok\expandafter{\@gls@short}%
```

Hook for further customisation if required:

```
6594 \glstrnewabbrevpresetkeyhook{#1}{#2}{#3}%
```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary. Ignore the category key (already obtained).

```
6595 \setkeys*{glstrabbrv}[category]{#1}%
```

Has the plural been explicitly set?

```
6596 \ifx\@gls@default@longpl\@gls@longpl  
6597 \else
```

Has the markwords attribute been set?

```
6598 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%  
6599 {%  
6600 \expandafter\@glstr@keywordseps\expandafter\@gls@longpl\expandafter  
6601 {\@gls@longpl}%  
6602 }%  
6603 {}%  
6604 \fi
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
6605 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%  
6606 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
6607 \@gls@setup@default@short@access{#3}%
```

Do any extra setup provided by hook:

```
6608 \newabbreviationhook
```

Define this entry:

```
6609 \protected@edef\@do@newglossaryentry{%
6610 \noexpand\newglossaryentry{\the\glslabeltok}%
6611 {%
6612 type=\glstrabbrvtype,%
6613 category=abbreviation,%
6614 short={\the\glsshorttok},%
6615 shortplural={\the\glsshortpltok},%
6616 long={\the\glslongtok},%
6617 longplural={\the\glslongpltok},%
6618 name={\the\glsshorttok},%
6619 \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
6620 \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
6621 \the\glskeylisttok
6622 }%
6623 }%
6624 \@do@newglossaryentry
6625 \GlsXtrPostNewAbbreviation
6626 }
```

**evpresetkeyhook** Hook for extra stuff in `\newabbreviation`

```
6627 \newcommand*{\glstrnewabbspresetkeyhook}[3]{}%
```

**NewAbbreviation** Hook used by abbreviation styles.

```
6628 \newcommand*{\GlsXtrPostNewAbbreviation}{}%
```

**bbreviationhook** Hook for use with `\newabbreviation`.

```
6629 \newcommand*{\newabbreviationhook}{}%
```

**reviationFields**

```
6630 \newcommand*{\CustomAbbreviationFields}{}%
```

**\glstrparen** For the parenthetical styles.

```
6631 \newcommand*{\glstrparen}[1]{(#1)}
```

**lsxtrfullformat** Full format without case change.

```
6632 \newcommand*{\glxtrfullformat}[2]{%
6633 \glsfirstlongfont{\glsaccesslong{#1}}#2\glxtrfullsep{#1}%
6634 \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6635 }
```

**lsxtrfullformat** Full format with case change.

```
6636 \newcommand*{\Glsxtrfullformat}[2]{%
6637 \glsfirstlongfont{\Glsaccesslong{#1}}#2\glxtrfullsep{#1}%
```

```

6638 \glxstrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6639 }

```

`\xtrfullplformat` Plural full format without case change.

```

6640 \newcommand*{\glxtrfullplformat}[2]{%
6641 \glsfirstlongfont{\glsaccesslongpl{#1}}#2\glxtrfullsep{#1}%
6642 \glxstrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}}}%
6643 }

```

`\xtrfullplformat` Plural full format with case change.

```

6644 \newcommand*{\Glsxtrfullplformat}[2]{%
6645 \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glxtrfullsep{#1}%
6646 \glxstrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6647 }

```

`\glxtrfullsep` Separator used by full format is a space by default. The argument is the entry's label.

```

6648 \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).

`\inlinefullformat` Full format without case change.

```

6649 \newcommand*{\glxtrininlinefullformat}{\glxtrfullformat}

```

`\inlinefullformat` Full format with case change.

```

6650 \newcommand*{\Glsxtrininlinefullformat}{\Glsxtrfullformat}

```

`\xtrfullplformat` Plural full format without case change.

```

6651 \newcommand*{\glxtrininlinefullplformat}{\glxtrfullplformat}

```

`\inefullplformat` Plural full format with case change.

```

6652 \newcommand*{\Glsxtrininlinefullplformat}{\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`

```

6653 \renewcommand*{\gl Sentryfull}[1]{\glxtrininlinefullformat{#1}{}}

```

`\Glsentryfull`

```

6654 \renewcommand*{\Glsentryfull}[1]{\Glsxtrininlinefullformat{#1}{}}

```

`\gl Sentryfullpl`

```

6655 \renewcommand*{\gl Sentryfullpl}[1]{\glxtrininlinefullplformat{#1}{}}

```

`\Glsentryfullpl`

```

6656 \renewcommand*{\Glsentryfullpl}[1]{\Glsxtrininlinefullplformat{#1}{}}

```

`\glsfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.  
6657 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.  
6658 `\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.  
6659 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont`  
6660 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.  
6661 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.  
6662 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\glsfirstlongfont` Font changing command used for the long form on first use or in the full format.  
6663 `\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}`

`\glsfirstlongdefaultfont`  
6664 `\newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}`

`\glsbrvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.  
6665 `\newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}`

`\glsbrvpluralsuffix` Default plural suffix.  
6666 `\newcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}`

`\glsxtrfull` Full form (no case-change).  
6667 `\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\@ns@glsxtrfull}`  
6668 `\newcommand*\@ns@glsxtrfull[2][{}]{%`  
6669 `\new@ifnextchar[{\@glsxtr@full{#1}{#2}}{%`  
6670 `{\@glsxtr@full{#1}{#2}[]}%`  
6671 `}`

`\@glsxtr@full` Low-level macro:  
6672 `\def\@glsxtr@full#1#2[#3]{%`  
If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).  
6673 `\@glsxtr@record{#1}{#2}{\glslink}%`  
6674 `\glsdoifexists{#2}%`  
6675 `{%`  
6676 `\glssetabbrvfmt{\glscategory{#2}}%`  
6677 `\let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper`  
6678 `\let\glsifplural\@secondoftwo`



```

6679 \let\glscapscase\@firstofthree
6680 \let\glsinsert\@empty
6681 \def\glscustomtext{\glsxtrinlinefullformat{#2}{#3}}%

```

What should `\glsxtrifwasfirstuse` be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

```

6682 \glsxtrsetupfulldefs
6683 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6684 }%
6685 \glspostlinkhook
6686 }

```

`trsetupfulldefs`

```

6687 \newcommand*{\glsxtrsetupfulldefs}{%
6688 \let\glsxtrifwasfirstuse\@firstoftwo
6689 }

```

`\Glsxtrfull` Full form (first letter uppercase).

```

6690 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\@ns@Glsxtrfull}
6691 \newcommand*\ns@Glsxtrfull[2][{}]{%
6692 \new@ifnextchar[\@Glsxtr@full{#1}{#2}}%
6693 {\@Glsxtr@full{#1}{#2}[]}%
6694 }

```

`\@Glsxtr@full` Low-level macro:

```

6695 \def\@Glsxtr@full#1#2[#3]{%
6696 \glsdoifexists{#2}%
6697 {%
6698 \glssetabbrvfmt{\glscategory{#2}}%
6699 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6700 \let\glsifplural\@secondoftwo
6701 \let\glscapscase\@secondofthree
6702 \let\glsinsert\@empty
6703 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6704 \glsxtrsetupfulldefs
6705 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6706 }%
6707 \glspostlinkhook
6708 }

```

`\GLSxtrfull` Full form (all uppercase).

```

6709 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\@ns@GLSxtrfull}
6710 \newcommand*\ns@GLSxtrfull[2][{}]{%
6711 \new@ifnextchar[\@GLSxtr@full{#1}{#2}}%
6712 {\@GLSxtr@full{#1}{#2}[]}%
6713 }

```

\@GLSxtr@full Low-level macro:

```
6714 \def\@GLSxtr@full#1#2[#3]{%
6715   \glsdoifexists{#2}%
6716   {%
6717     \glsetabbrvfmt{\glscategory{#2}}%
6718     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6719     \let\glsifplural\@secondoftwo
6720     \let\glscapscase\@thirdofthree
6721     \let\glsinsert\@empty
6722     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
6723     \glsxtrsetupfulldefs
6724     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6725   }%
6726   \glspostlinkhook
6727 }
```

\glsxtrfullpl Plural full form (no case-change).

```
6728 \newrobustcmd*{\glsxtrfullpl}{\@gl@hyp@opt\@ns@glsxtrfullpl}
6729 \newcommand*\ns@glsxtrfullpl[2][ ]{%
6730   \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}%
6731   {\@glsxtr@fullpl{#1}{#2}[ ]}%
6732 }
```

\@glsxtr@fullpl Low-level macro:

```
6733 \def\@glsxtr@fullpl#1#2[#3]{%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
6734   \@glsxtr@record{#1}{#2}{glslink}%
6735   \glsdoifexists{#2}%
6736   {%
6737     \glsetabbrvfmt{\glscategory{#2}}%
6738     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6739     \let\glsifplural\@firstoftwo
6740     \let\glscapscase\@firstofthree
6741     \let\glsinsert\@empty
6742     \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
6743     \glsxtrsetupfulldefs
6744     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6745   }%
6746   \glspostlinkhook
6747 }
```

\Glsxtrfullpl Plural full form (first letter uppercase).

```
6748 \newrobustcmd*{\Glsxtrfullpl}{\@gl@hyp@opt\@ns@Glsxtrfullpl}
6749 \newcommand*\ns@Glsxtrfullpl[2][ ]{%
6750   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
6751   {\@Glsxtr@fullpl{#1}{#2}[ ]}%
6752 }
```

\@Glsxtr@fullpl Low-level macro:

```
6753 \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).

```
6754 \@glxtr@record{#1}{#2}{glslink}%
6755 \glstoifexists{#2}%
6756 {%
6757 \glsetabbrvfmt{\glscategory{#2}}%
6758 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6759 \let\gl@ifplural\@firstoftwo
6760 \let\glscapscase\@secondofthree
6761 \let\glinsert\@empty
6762 \def\glscustomtext{\@Glsxtrinlinefullplformat{#2}{#3}}%
6763 \glxtrsetupfulldefs
6764 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6765 }%
6766 \glspostlinkhook
6767 }
```

\GLSxtrfullpl Plural full form (all upper case).

```
6768 \newrobustcmd*{\GLSxtrfullpl}{\@gl@hyp@opt\@ns@GLSxtrfullpl}
6769 \newcommand*\ns@GLSxtrfullpl[2][{}]{%
6770 \new@ifnextchar[\@GLSxtr@fullpl{#1}{#2}}%
6771 {\@GLSxtr@fullpl{#1}{#2}[]}%
6772 }
```

\@GLSxtr@fullpl Low-level macro:

```
6773 \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).

```
6774 \@glxtr@record{#1}{#2}{glslink}%
6775 \glstoifexists{#2}%
6776 {%
6777 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6778 \let\gl@ifplural\@firstoftwo
6779 \let\glscapscase\@thirdofthree
6780 \let\glinsert\@empty
6781 \def\glscustomtext{%
6782 \mfirstucMakeUppercase{\@glxtrinlinefullplformat{#2}{#3}}}%
6783 \glxtrsetupfulldefs
6784 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6785 }%
6786 \glspostlinkhook
6787 }
```

The short and long forms work in a similar way to acronyms.

`\glxtrshort`

```
6788 \newrobustcmd*{\glxtrshort}{\@gls@hyp@opt\ns@glxtrshort}
```

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

```
6789 \newcommand*{\ns@glxtrshort}[2] [] {%
```

```
6790   \new@ifnextchar[{\@glxtrshort{#1}{#2}}{\@glxtrshort{#1}{#2} []}%
```

```
6791 }
```

Read in the final optional argument:

```
6792 \def\@glxtrshort#1#2[#3] {%
```

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

```
6793   \@glxtr@record{#1}{#2}{glslink}%
```

```
6794   \glsdoifexists{#2}%
```

```
6795   {%
```

Need to make sure `\glsabbrvfont` is set correctly.

```
6796     \glssetabrvfmt{\glscategory{#2}}%
```

```
6797     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
6798     \let\glxtrifwasfirstuse\@secondoftwo
```

```
6799     \let\glsifplural\@secondoftwo
```

```
6800     \let\glscapscase\@firstofthree
```

```
6801     \let\glsinsert\@empty
```

```
6802     \def\glscustomtext{%
```

```
6803       \glsabbrvfont{\glsaccessshort{#2}\ifglxtrininsertinside#3\fi}%
```

```
6804       \ifglxtrininsertinside\else#3\fi
```

```
6805     }%
```

```
6806     \@gls@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
```

```
6807   }%
```

```
6808   \glspostlinkhook
```

```
6809 }
```

`\Glsxtrshort`

```
6810 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\ns@Glsxtrshort}
```

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

```
6811 \newcommand*{\ns@Glsxtrshort}[2] [] {%
```

```
6812   \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
```

```
6813 }
```

Read in the final optional argument:

```
6814 \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).

```
6815   \@glxtr@record{#1}{#2}{glslink}%
```

```
6816   \glsdoifexists{#2}%
```

```
6817   {%
```

```
6818     \glssetabrvfmt{\glscategory{#2}}%
```

```
6819     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```

6820 \let\glxtrifwasfirstuse\@secondoftwo
6821 \let\glxifplural\@secondoftwo
6822 \let\glscapscase\@secondofthree
6823 \let\glxinsert\@empty
6824 \def\glscustomtext{%
6825     \glxabbrvfont{\glxaccessshort{#2}\ifglxtrinsertinside#3\fi}%
6826     \ifglxtrinsertinside\else#3\fi
6827 }%
6828 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6829 }%
6830 \glxpostlinkhook
6831 }

```

\GLSxtrshort

```

6832 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\@ns@GLSxtrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
6833 \newcommand*{\ns@GLSxtrshort}[2] [] {%
6834     \new@ifnextchar[\@GLSxtrshort{#1}{#2}]{\@GLSxtrshort{#1}{#2} []}%
6835 }

```

Read in the final optional argument:

```

6836 \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).
6837     \@glxtr@record{#1}{#2}{glxlink}%
6838     \glxdoifexists{#2}%
6839     {%
6840         \glxsetabbrvfmt{\glxcategory{#2}}%
6841         \let\do@glx@link@checkfirsthyper\@glx@link@nocheckfirsthyper
6842         \let\glxtrifwasfirstuse\@secondoftwo
6843         \let\glxifplural\@secondoftwo
6844         \let\glscapscase\@thirdofthree
6845         \let\glxinsert\@empty
6846         \def\glscustomtext{%
6847             \mfirstucMakeUppercase
6848             {\glxabbrvfont{\glxaccessshort{#2}\ifglxtrinsertinside#3\fi}%
6849             \ifglxtrinsertinside\else#3\fi
6850         }%
6851     }%
6852     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6853 }%
6854 \glxpostlinkhook
6855 }

```

\glxtrlong

```

6856 \newrobustcmd*{\glxtrlong}{\@gls@hyp@opt\@ns@glxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument

```

```

6857 \newcommand*{\ns@glxtrlong}[2] [] {%
6858   \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
6859 }

```

Read in the final optional argument:

```

6860 \def\@glxtrlong#1#2[#3]{%

```

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

```

6861   \@glxtr@record{#1}{#2}{glslink}%
6862   \glsoifexists{#2}%
6863   {%
6864     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6865     \let\glxtrifwasfirstuse\@secondoftwo
6866     \let\gl@ifplural\@secondoftwo
6867     \let\glscapscase\@firstofthree
6868     \let\glinsert\@empty
6869     \def\glscustomtext{%
6870       \glslongfont{\glaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6871       \ifglxtrininsertinside\else#3\fi
6872     }%
6873     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6874   }%
6875   \glspostlinkhook
6876 }

```

\Glsxtrlong

```

6877 \newrobustcmd*{\Glsxtrlong}{\@gl@hyp@opt\ns@Glsxtrlong}

```

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

```

6878 \newcommand*{\ns@Glsxtrlong}[2] [] {%
6879   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
6880 }

```

Read in the final optional argument:

```

6881 \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).

```

6882   \@glxtr@record{#1}{#2}{glslink}%
6883   \glsoifexists{#2}%
6884   {%
6885     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6886     \let\glxtrifwasfirstuse\@secondoftwo
6887     \let\gl@ifplural\@secondoftwo
6888     \let\glscapscase\@secondofthree
6889     \let\glinsert\@empty
6890     \def\glscustomtext{%
6891       \glslongfont{\Glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6892       \ifglxtrininsertinside\else#3\fi
6893     }%

```

```

6894 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6895 }%
6896 \glspostlinkhook
6897 }

```

\GLSxtrlong

```

6898 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\@ns@GLSxtrlong}

Define the un-starred form. Need to determine if there is a final optional argument
6899 \newcommand*{\ns@GLSxtrlong}[2] [] {%
6900 \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
6901 }

Read in the final optional argument:
6902 \def\@GLSxtrlong#1#2[#3] {%

If the record option has been used, the information needs to be written to the aux file regard-
less of whether the entry exists (unless indexing has been switched off).
6903 \@glsxtr@record{#1}{#2}{glslink}%
6904 \glsdoifexists{#2}%
6905 {%
6906 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6907 \let\glsxtrifwasfirstuse\@secondoftwo
6908 \let\glsifplural\@secondoftwo
6909 \let\gls caps case\@thirdofthree
6910 \let\glsinsert\@empty
6911 \def\gls custom text{%
6912 \mfirstucMakeUppercase
6913 {\gls long font{\gls access long{#2}\ifglsxtrinsertinside#3\fi}%
6914 \ifglsxtrinsertinside\else#3\fi
6915 }%
6916 }%
6917 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6918 }%
6919 \gls post link hook
6920 }

```

Plural short forms:

\glsxtrshortpl

```

6921 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\@ns@glsxtrshortpl}

Define the un-starred form. Need to determine if there is a final optional argument
6922 \newcommand*{\ns@glsxtrshortpl}[2] [] {%
6923 \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
6924 }

Read in the final optional argument:
6925 \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).

```

6926 \@glstr@record{#1}{#2}{glslink}%
6927 \glsoifexists{#2}%
6928 {%
6929 \glsetabbrvfmt{\glscategory{#2}}%
6930 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6931 \let\glstrifwasfirstuse\@secondoftwo
6932 \let\gl@ifplural\@firstoftwo
6933 \let\glscapscase\@firstofthree
6934 \let\glinsert\@empty
6935 \def\glscustomtext{%
6936 \glabbrvfont{\glaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
6937 \ifglstrinsertinside\else#3\fi
6938 }%
6939 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6940 }%
6941 \glspostlinkhook
6942 }

```

\Glsxtrshortpl

```

6943 \newrobustcmd*{\Glsxtrshortpl}{\@gl@hyp@opt\ns@Glsxtrshortpl}

```

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

```

6944 \newcommand*{\ns@Glsxtrshortpl}[2][{}]{%
6945 \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2}[]}%
6946 }

```

Read in the final optional argument:

```

6947 \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).

```

6948 \@glstr@record{#1}{#2}{glslink}%
6949 \glsoifexists{#2}%
6950 {%
6951 \glsetabbrvfmt{\glscategory{#2}}%
6952 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6953 \let\glstrifwasfirstuse\@secondoftwo
6954 \let\gl@ifplural\@firstoftwo
6955 \let\glscapscase\@secondofthree
6956 \let\glinsert\@empty
6957 \def\glscustomtext{%
6958 \glabbrvfont{\Glsaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
6959 \ifglstrinsertinside\else#3\fi
6960 }%
6961 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6962 }%
6963 \glspostlinkhook
6964 }

```



\GLSxtrshortpl

6965 \newrobustcmd\*{\GLSxtrshortpl}{\@gls@hyp@opt\@ns@GLSxtrshortpl}

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

6966 \newcommand\*{\ns@GLSxtrshortpl}[2] [] {%  
6967 \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%  
6968 }

Read in the final optional argument:

6969 \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).

6970 \@glsxtr@record{#1}{#2}{glslink}%  
6971 \glsdoifexists{#2}%  
6972 {%  
6973 \glssetabbrvfmt{\gls@category{#2}}%  
6974 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
6975 \let\glsxtrifwasfirstuse\@secondoftwo  
6976 \let\glsifplural\@firstoftwo  
6977 \let\gls@scapscase\@thirdofthree  
6978 \let\glsinsert\@empty  
6979 \def\gls@customtext{%  
6980 \mfirstucMakeUppercase  
6981 {\glsabbrvfont{\gls@accessshortpl{#2}\ifglsxtrininsertinside#3\fi}%  
6982 \ifglsxtrininsertinside\else#3\fi  
6983 }%  
6984 }%  
6985 \@gls@link[#1]{#2}{\csname gls@\gls@type @entryfmt\endcsname}%  
6986 }%  
6987 \gls@postlinkhook  
6988 }

Plural long forms:

\glsxtrlongpl

6989 \newrobustcmd\*{\glsxtrlongpl}{\@gls@hyp@opt\@ns@glsxtrlongpl}

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

6990 \newcommand\*{\ns@glsxtrlongpl}[2] [] {%  
6991 \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2} []}%  
6992 }

Read in the final optional argument:

6993 \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).

6994 \@glsxtr@record{#1}{#2}{glslink}%  
6995 \glsdoifexists{#2}%  
6996 {%

```

6997 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6998 \let\glxtrifwasfirstuse\@secondoftwo
6999 \let\gl@sifplural\@firstoftwo
7000 \let\glscapscase\@firstofthree
7001 \let\gl$insert\@empty
7002 \def\glscustomtext{%
7003     \glslongfont{\gl@saccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
7004     \ifglxtrininsertinside\else#3\fi
7005 }%
7006 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7007 }%
7008 \glspostlinkhook
7009 }

```

\Glsxtrlongpl

```

7010 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\@ns@Glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7011 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
7012     \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
7013 }
    Read in the final optional argument:
7014 \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).
7015     \glxtr@record{#1}{#2}{glslink}%
7016     \glsdoidexists{#2}%
7017     {%
7018         \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7019         \let\glxtrifwasfirstuse\@secondoftwo
7020         \let\gl@sifplural\@firstoftwo
7021         \let\glscapscase\@secondofthree
7022         \let\gl$insert\@empty
7023         \def\glscustomtext{%
7024             \glslongfont{\Glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
7025             \ifglxtrininsertinside\else#3\fi
7026         }%
7027         \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7028     }%
7029     \glspostlinkhook
7030 }

```

\GLSxtrlongpl

```

7031 \newrobustcmd*{\GLSxtrlongpl}{\@gl@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7032 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
7033     \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
7034 }

```

Read in the final optional argument:

```
7035 \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).

```
7036 \@glstr@record{#1}{#2}{glslink}%
7037 \glsoifexists{#2}%
7038 {%
7039   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7040   \let\glstrifwasfirstuse\@secondoftwo
7041   \let\gl@ifplural\@firstoftwo
7042   \let\glscapscase\@thirdofthree
7043   \let\glinsert\@empty
7044   \def\glscustomtext{%
7045     \mfirstucMakeUppercase
7046     {\glslongfont{\glaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
7047     \ifglstrinsertinside\else#3\fi
7048   }%
7049   }%
7050   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7051 }%
7052 \glspostlinkhook
7053 }
```

**\glsetabbrvfmt** Set the current format for the given category (or the abbreviation category if unset).

```
7054 \newcommand*{\glsetabbrvfmt}[1]{%
7055   \ifcsdef{@glabbrv@current@#1}%
7056   {\glstr@applyabbrvfmt{\csname @glabbrv@current@#1\endcsname}}%
7057   {\glstr@applyabbrvfmt{\@glabbrv@current@abbreviation}}%
7058 }
```

**glsuseabbrvfont** Provide a way to use the abbreviation font for a given category for arbitrary text.

```
7059 \newrobustcmd*{\gluseabbrvfont}[2]{\glsetabbrvfmt{#2}\glabbrvfont{#1}}
```

**\glsuselongfont** Provide a way to use the long font for a given category for arbitrary text.

```
7060 \newrobustcmd*{\glsuselongfont}[2]{\glsetabbrvfmt{#2}\glslongfont{#1}}
```

**sxtrgenabbrvfmt** Similar to \glsgenacfmt, but for abbreviations.

```
7061 \newcommand*{\glsxtrgenabbrvfmt}{%
7062   \ifdefempty\glscustomtext
7063   {%
7064     \ifglused\glslabel
7065     {%
```

Subsequent use:

```
7066   \gl@ifplural
7067   {%
```

Subsequent plural form:

```
7068      \glscapscase
7069      {%
```

Subsequent plural form, don't adjust case:

```
7070      \glxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7071      }%
7072      {%
```

Subsequent plural form, make first letter upper case:

```
7073      \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7074      }%
7075      {%
```

Subsequent plural form, all caps:

```
7076      \mfirstucMakeUppercase
7077      {\glxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
7078      }%
7079      }%
7080      {%
```

Subsequent singular form

```
7081      \glscapscase
7082      {%
```

Subsequent singular form, don't adjust case:

```
7083      \glxtrsubsequentfmt{\glslabel}{\glsinsert}%
7084      }%
7085      {%
```

Subsequent singular form, make first letter upper case:

```
7086      \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
7087      }%
7088      {%
```

Subsequent singular form, all caps:

```
7089      \mfirstucMakeUppercase
7090      {\glxtrsubsequentfmt{\glslabel}{\glsinsert}}%
7091      }%
7092      }%
7093      }%
7094      {%
```

First use:

```
7095      \glsifplural
7096      {%
```

First use plural form:

```
7097      \glscapscase
7098      {%
```

First use plural form, don't adjust case:

```
7099      \glxtrfullplformat{\glslabel}{\glsinsert}%
```

7100 }%

7101 {%

First use plural form, make first letter upper case:

7102 \Glsxtrfullplformat{\glslabel}{\glsinsert}%

7103 }%

7104 {%

First use plural form, all caps:

7105 \mfirstucMakeUppercase

7106 {\glsxtrfullplformat{\glslabel}{\glsinsert}}%

7107 }%

7108 }%

7109 {%

First use singular form

7110 \glscapscase

7111 {%

First use singular form, don't adjust case:

7112 \glsxtrfullformat{\glslabel}{\glsinsert}%

7113 }%

7114 {%

First use singular form, make first letter upper case:

7115 \Glsxtrfullformat{\glslabel}{\glsinsert}%

7116 }%

7117 {%

First use singular form, all caps:

7118 \mfirstucMakeUppercase

7119 {\glsxtrfullformat{\glslabel}{\glsinsert}}%

7120 }%

7121 }%

7122 }%

7123 }%

7124 {%

User supplied text.

7125 \glscustomtext

7126 }%

7127 }

trsubsequentfmt Subsequent use format (singular no case change).

7128 \newcommand\*{\glsxtrsubsequentfmt}[2]{%

7129 \glsabbrvfont{\glsaccessshort{#1}}\ifglsxtrininsertinside #2\fi}%

7130 \ifglsxtrininsertinside \else#2\fi

7131 }

7132 \let\glsxtrdefaultsubsequentfmt\glsxtrsubsequentfmt

subsequentplfmt Subsequent use format (plural no case change).

7133 \newcommand\*{\glsxtrsubsequentplfmt}[2]{%

```

7134 \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7135 \ifglxtrinsertinside \else#2\fi
7136 }
7137 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).

```

7138 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7139 \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7140 \ifglxtrinsertinside \else#2\fi
7141 }
7142 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

```

subsequentplfmt Subsequent use format (plural, first letter uppercase).

```

7143 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
7144 \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
7145 \ifglxtrinsertinside \else#2\fi
7146 }
7147 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

### 1.7.1 Abbreviation Styles Setup

breviationstyle

```

7148 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7149 \ifcsundef{@glsabbrv@dispstyle@setup@#2}
7150 {%
7151 \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7152 }%
7153 {%

```

Have abbreviations already been defined for this category?

```

7154 \ifcsstring{@glsabbrv@current@#1}{#2}%
7155 {%

```

Style already set.

```

7156 }%
7157 {%
7158 \def\@glxtr@dostylewarn{%
7159 \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7160 {%
7161 \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
7162 style has been switched \MessageBreak
7163 for category ‘#1’, \MessageBreak
7164 but there have already been entries \MessageBreak
7165 defined for this category. Unwanted \MessageBreak
7166 side-effects may result}}%
7167 \@endfortrue
7168 }%
7169 \@glxtr@dostylewarn

```

Set up the style for the given category.

```

7170      \csdef{@glsabbrv@current@#1}{#2}%
7171      \glsxtr@applyabbrvstyle{#2}%
7172      }%
7173  }%
7174 }
```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```

7175 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
7176   \csuse{@glsabbrv@dispstyle@setup@#1}%
7177   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7178 }
```

`\r@applyabbrvfmt` Only apply the style formats.

```

7179 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
7180   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7181 }
```

`\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.

```

7182 \newcommand*{\newabbreviationstyle}[3]{%
7183   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7184   {%
7185     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7186       defined}{}%
7187   }%
7188   {%
7189     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

7190     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7191     #2}%
7192     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

7193     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7194     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7195     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7196     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glsxtrsubsequentfmt` etc in case a style changes this.

```

7197     \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
7198     \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
7199     \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7200     \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7201     #3}%
7202   }%
7203 }
```

abbreviationstyle

```

7204 \newcommand*{\renewabbreviationstyle}[3]{%
7205   \ifcsundef{@glsabbrv@dispstyle@setup@#1}
7206   {%
7207     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7208   }%
7209   {%
7210     \csdef{@glsabbrv@dispstyle@setup@#1}{%
      Initialise hook to do nothing. The style may change this.
7211       \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7212       #2}%
7213     \csdef{@glsabbrv@dispstyle@fmts@#1}{%
      Assume in-line form is the same as first use. The style may change this.
7214       \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7215       \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7216       \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7217       \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
7218       #3}%
7219     }%
7220 }

```

abbreviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style’s name.

```

7221 \newcommand*{\letabbreviationstyle}[2]{%
7222   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7223   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7224 }

```

deprecated@abbrstyle

```
\glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}
```

Define a synonym for a deprecated abbreviation style.

```

7225 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
7226   \csdef{@glsabbrv@dispstyle@setup@#1}{%
7227     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7228     \csuse{@glsabbrv@dispstyle@setup@#2}%
7229   }%
7230   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7231 }

```

deprecatedAbbrStyle Generate warning for deprecated style use.

```

7232 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7233   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7234   use ‘#2’ instead}%
7235 }

```



eAbbrStyleSetup

```

7236 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7237   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
7238   {%
7239     \PackageError{glossaries-extra}%
7240     {Unknown abbreviation style definitions ‘#1’}{}%
7241   }%
7242   {%
7243     \csname @glsabbrv@dispstyle@setup@#1\endcsname
7244   }%
7245 }

```

seAbbrStyleFmts

```

7246 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
7247   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
7248   {%
7249     \PackageError{glossaries-extra}%
7250     {Unknown abbreviation style formats ‘#1’}{}%
7251   }%
7252   {%
7253     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
7254   }%
7255 }

```

### 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.

```

7256 \newif\ifglsxtrinsertinside
7257 \glsxtrinsertinsidefalse

```

trlongshortname

```

7258 \newcommand*{\glsxtrlongshortname}{%
7259   \protect\glsabbrvfont{\the\glsshorttok}%
7260 }

```

long-short

```

7261 \newabbreviationstyle{long-short}%
7262 {%

```

```

7263 \renewcommand*{\CustomAbbreviationFields}{%
7264   name={\glxtrlongshortname},
7265   sort={\the\glsshorttok},
7266   first={\protect\glsfirstlongfont{\the\glslongtok}%
7267     \protect\glxtrfullsep{\the\glslabeltok}%
7268     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7269   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7270     \protect\glxtrfullsep{\the\glslabeltok}%
7271     \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
7272   plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
7273   description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

7274 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7275   \glshasattribute{\the\glslabeltok}{regular}%
7276   {%
7277     \glssetattribute{\the\glslabeltok}{regular}{false}%
7278   }%
7279   {}%
7280 }%
7281}%
7282{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7283 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7284 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7285 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7286 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7287 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7288 \renewcommand*{\glxtrfullformat}[2]{%
7289   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7290   \ifglxtrininsertinside\else##2\fi
7291   \glxtrfullsep{##1}%
7292   \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7293 }%
7294 \renewcommand*{\glxtrfullplformat}[2]{%
7295   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7296   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7297   \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7298 }%
7299 \renewcommand*{\Glsxtrfullformat}[2]{%
7300   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7301   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7302   \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
7303 }%
7304 \renewcommand*{\Glsxtrfullplformat}[2]{%
7305   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7306   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

7307 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7308 }%
7309 }

```

Set this as the default style for general abbreviations:

```

7310 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

7311 \newcommand*{\glsxtrlongshortdescsort}{%
7312 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
7313 }

```

ngshortdescname

```

7314 \newcommand*{\glsxtrlongshortdescname}{%
7315 \protect\glslongfont{\the\glslongtok}
7316 \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
7317 }

```

long-short-desc User supplies description. The long form is included in the name.

```

7318 \newabbreviationstyle{long-short-desc}%
7319 {%
7320 \renewcommand*{\CustomAbbreviationFields}{%
7321 name={\glsxtrlongshortdescname},
7322 sort={\glsxtrlongshortdescsort},%
7323 first={\protect\glsfirstlongfont{\the\glslongtok}%
7324 \protect\glsxtrfullsep{\the\glslabeltok}%
7325 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7326 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7327 \protect\glsxtrfullsep{\the\glslabeltok}%
7328 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

7329 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7330 plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
7331 }%

```

Unset the regular attribute if it has been set.

```

7332 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7333 \glshasattribute{\the\glslabeltok}{regular}%
7334 {%
7335 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7336 }%
7337 {}%
7338 }%
7339 }%
7340 {%
7341 \GlsXtrUseAbbrStyleFmts{long-short}%
7342 }

```

trshortlongname

```
7343 \newcommand*{\glxtrshortlongname}{%
7344   \protect\glsabbrvfont{\the\glsshorttok}%
7345 }
```

short-long Short form followed by long form in parenthesis on first use.

```
7346 \newabbreviationstyle{short-long}%
7347 {%
7348   \renewcommand*{\CustomAbbreviationFields}{%
7349     name={\glxtrshortlongname},
7350     sort={\the\glsshorttok},
7351     description={\the\glslongtok},%
7352     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
7353       \protect\glxtrfullsep{\the\glslabeltok}%
7354       \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7355     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7356       \protect\glxtrfullsep{\the\glslabeltok}%
7357       \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7358     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7359 }
```

Unset the regular attribute if it has been set.

```
7359 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7360   \glshasattribute{\the\glslabeltok}{regular}%
7361   {%
7362     \glissetattribute{\the\glslabeltok}{regular}{false}%
7363   }%
7364   {}%
7365 }%
7366 }%
7367 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7368 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7369 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7370 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7371 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7372 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7373 \renewcommand*{\glxtrfullformat}[2]{%
7374   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7375   \ifglxtrininsertinside\else##2\fi
7376   \glxtrfullsep{##1}%
7377   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7378 }%
7379 \renewcommand*{\glxtrfullplformat}[2]{%
7380   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7381   \ifglxtrininsertinside\else##2\fi
7382   \glxtrfullsep{##1}%
7383 }
```

```

7383 \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7384 }%
7385 \renewcommand*{\Glsxtrfullformat}[2]{%
7386 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
7387 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7388 \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7389 }%
7390 \renewcommand*{\Glsxtrfullplformat}[2]{%
7391 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
7392 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7393 \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7394 }%
7395 }

```

ortlongdescsort

```

7396 \newcommand*{\glsxtrshortlongdescsort}{\the\glsshorttok}

```

ortlongdescname

```

7397 \newcommand*{\glsxtrshortlongdescname}{%
7398 \protect\glsabbrvfont{\the\glsshorttok}
7399 \glsxtrparen{\protect\glslongfont{\the\glslongtok}}}%
7400 }

```

short-long-desc User supplies description. The long form is included in the name.

```

7401 \newabbreviationstyle{short-long-desc}%
7402 {%
7403 \renewcommand*{\CustomAbbreviationFields}{%
7404 name={\glsxtrshortlongdescname},
7405 sort={\glsxtrshortlongdescsort},
7406 first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
7407 \protect\glsxtrfullsep{\the\glslabeltok}}%
7408 \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
7409 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
7410 \protect\glsxtrfullsep{\the\glslabeltok}}%
7411 \glsxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
7412 text={\protect\glsabbrvfont{\the\glsshorttok}},%
7413 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
7414 }%

```

Unset the regular attribute if it has been set.

```

7415 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7416 \glshasattribute{\the\glslabeltok}{regular}%
7417 {%
7418 \glissetattribute{\the\glslabeltok}{regular}{false}%
7419 }%
7420 {}%
7421 }%

```

```

7422 }%
7423 {%
7424   \GlsXtrUseAbbrStyleFmts{short-long}%
7425 }

```

`\longfootnotefont` Only used by the “footnote” styles.

```

7426 \newcommand*{\glfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%

```

`\longfootnotefont` Only used by the “footnote” styles.

```

7427 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%

```

`\glsextrabrvfootnote` `\glsextrabrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument `<long>` includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```

7428 \newcommand*{\glsextrabrvfootnote}[2]{\footnote{#2}}

```

`\glsextrfootnotename`

```

7429 \newcommand*{\glsextrfootnotename}{%
7430   \protect\glsabbrvfont{the\glsshorttok}%
7431 }

```

`\footnote` Short form followed by long form in footnote on first use.

```

7432 \newabbreviationstyle{footnote}%
7433 {%
7434   \renewcommand*{\CustomAbbreviationFields}{%
7435     name={\glsextrfootnotename},
7436     sort={the\glsshorttok},
7437     description={the\glslongtok},%

7438     first={\protect\glfirstabbrvfont{the\glsshorttok}}%
7439     \protect\glsextrabrvfootnote{the\glslabeltok}%
7440     {\protect\glfirstlongfootnotefont{the\glslongtok}}},%
7441     firstplural={\protect\glfirstabbrvfont{the\glsshortpltok}}%
7442     \protect\glsextrabrvfootnote{the\glslabeltok}%
7443     {\protect\glfirstlongfootnotefont{the\glslongpltok}}},%

7444     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.

```

7445 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7446   \glsssetattribute{the\glslabeltok}{nohyperfirst}{true}%
7447   \glshasattribute{the\glslabeltok}{regular}%

```

```

7448   {%
7449     \glssetattribute{\the\glslabeltok}{regular}{false}%
7450   }%
7451   {}%
7452 }%
7453 }%
7454 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7455 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
7456 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7457 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7458 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7459 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7460 \renewcommand*\glsxtrfullformat[2]{%
7461   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7462   \ifglsxtrininsertinside\else##2\fi
7463   \protect\glsxtrabbrvfootnote{##1}%
7464   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7465 }%
7466 \renewcommand*\glsxtrfullplformat[2]{%
7467   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7468   \ifglsxtrininsertinside\else##2\fi
7469   \protect\glsxtrabbrvfootnote{##1}%
7470   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7471 }%
7472 \renewcommand*\Glsxtrfullformat[2]{%
7473   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7474   \ifglsxtrininsertinside\else##2\fi
7475   \protect\glsxtrabbrvfootnote{##1}%
7476   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7477 }%
7478 \renewcommand*\Glsxtrfullplformat[2]{%
7479   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7480   \ifglsxtrininsertinside\else##2\fi
7481   \protect\glsxtrabbrvfootnote{##1}%
7482   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7483 }%

```

The first use full form and the inline full form use the short (long) style.

```

7484 \renewcommand*\glsxtrinlinefullformat[2]{%
7485   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7486   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7487   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7488 }%
7489 \renewcommand*\glsxtrinlinefullplformat[2]{%
7490   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7491   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7492   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%

```

```

7493 }%
7494 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7495   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7496   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7497   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7498 }%
7499 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7500   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7501   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7502   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7503 }%
7504 }

```

#### short-footnote

```
7505 \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.

```

7506 \newabbreviationstyle{postfootnote}%
7507 {%
7508   \renewcommand*{\CustomAbbreviationFields}{%
7509     name={\glxtrfootnotename},
7510     sort={\the\glsshorttok},
7511     description={\the\glslongtok},%
7512     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7513     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7514     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.

```

7515   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7516     \csdef{glxtrpostlink\glscategorylabel}{%
7517       \glxtrifwasfirstuse
7518       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7519         \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7520         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}%
7521       }%
7522     }%
7523   }%
7524   \glshasattribute{\the\glslabeltok}{regular}%
7525   {%
7526     \glssetattribute{\the\glslabeltok}{regular}{false}%
7527   }%
7528   {}%
7529 }%

```



The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
7530 \renewcommand*{\glxtrsetupfulldefs}{%
7531   \let\glxtrifwasfirstuse\@secondoftwo
7532 }%
7533 }%
7534 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7535 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7536 \renewcommand*{\glsabrvfont}[1]{\glsabrvdefaultfont{##1}}%
7537 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7538 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
7539 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
7540 \renewcommand*{\glxtrfullformat}[2]{%
7541   \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7542   \ifglxtrininsertinside\else##2\fi
7543 }%
7544 \renewcommand*{\glxtrfullplformat}[2]{%
7545   \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7546   \ifglxtrininsertinside\else##2\fi
7547 }%
7548 \renewcommand*{\Glsxtrfullformat}[2]{%
7549   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7550   \ifglxtrininsertinside\else##2\fi
7551 }%
7552 \renewcommand*{\Glsxtrfullplformat}[2]{%
7553   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7554   \ifglxtrininsertinside\else##2\fi
7555 }%
```

The first use full form and the inline full form use the short (long) style.

```
7556 \renewcommand*{\glxtrinlinefullformat}[2]{%
7557   \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7558   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7559   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
7560 }%
7561 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7562   \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7563   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7564   \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
7565 }%
7566 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7567   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7568   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7569   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7570 }%
7571 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%

```

```

7572 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7573 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7574 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7575 }%
7576 }

```

rt-postfootnote

```

7577 \letabbreviationstyle{short-postfootnote}{postfootnote}

```

shortnolongname

```

7578 \newcommand*{\glxtrshortnolongname}{%
7579 \protect\glsabbrvfont{\the\glsshorttok}%
7580 }

```

**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.

```

7581 \newabbreviationstyle{short}%
7582 {%
7583 \renewcommand*{\CustomAbbreviationFields}{%
7584 name={\glxtrshortnolongname},
7585 sort={\the\glsshorttok},
7586 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7587 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7588 text={\protect\glsabbrvfont{\the\glsshorttok}},
7589 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7590 description={\the\glslongtok}}%
7591 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7592 \glssetattribute{\the\glslabeltok}{regular}{true}}%
7593 }%
7594 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7595 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7596 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7597 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7598 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7599 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7600 \renewcommand*{\glxtrinlinefullformat}[2]{%
7601 \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7602 \ifglxtrinsertinside##2\fi}%
7603 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7604 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7605 }%
7606 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7607 \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%

```

```

7608     \ifglxtrinsertinside##2\fi}%
7609     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7610     \glxtrparen{\glsfirslongfont{\glssaccesslongpl{##1}}}%
7611 }%
7612 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7613     \protect\glsfirstabbrvfont{\glssaccessshort{##1}%
7614         \ifglxtrinsertinside##2\fi}%
7615     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7616     \glxtrparen{\glsfirslongfont{\Glsaccesslong{##1}}}%
7617 }%
7618 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7619     \protect\glsfirstabbrvfont{\glssaccessshortpl{##1}%
7620         \ifglxtrinsertinside##2\fi}%
7621     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7622     \glxtrparen{\glsfirslongfont{\Glsaccesslongpl{##1}}}%
7623 }%

```

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.

```

7624 \renewcommand*{\glxtrfullformat}[2]{%
7625     \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7626     \ifglxtrinsertinside\else##2\fi
7627 }%
7628 \renewcommand*{\glxtrfullplformat}[2]{%
7629     \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7630     \ifglxtrinsertinside\else##2\fi
7631 }%
7632 \renewcommand*{\Glsxtrfullformat}[2]{%
7633     \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7634     \ifglxtrinsertinside\else##2\fi
7635 }%
7636 \renewcommand*{\Glsxtrfullplformat}[2]{%
7637     \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7638     \ifglxtrinsertinside\else##2\fi
7639 }%
7640 }

```

Set this as the default style for acronyms:

```

7641 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

7642 \letabbreviationstyle{short-nolong}{short}

```

short-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

7643 \newabbreviationstyle{short-nolong-noreg}%
7644 {%
7645     \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

7646 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

7647 \glshasattribute{\the\glslabeltok}{regular}%
7648 {%
7649 \glissetattribute{\the\glslabeltok}{regular}{false}%
7650 }%
7651 {}%
7652 }%
7653 }%
7654 {%
7655 \GlsXtrUseAbbrStyleFmts{short-nolong}%
7656 }

```

trshortdescname

```

7657 \newcommand*{\glxtrshortdescname}{%
7658 \protect\glsabbrvfont{\the\glsshorttok}%
7659 }

```

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.

```

7660 \newabbreviationstyle{short-desc}%
7661 {%
7662 \renewcommand*{\CustomAbbreviationFields}{%
7663 name={\glxtrshortdescname},
7664 sort={\the\glsshorttok},
7665 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7666 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7667 text={\protect\glsabbrvfont{\the\glsshorttok}},
7668 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7669 description={\the\glslongtok}}%
7670 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7671 \glissetattribute{\the\glslabeltok}{regular}{true}}%
7672 }%
7673 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7674 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7675 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7676 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7677 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7678 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7679 \renewcommand*{\glxtrinlinefullformat}[2]{%
7680 \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
7681 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7682 \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7683 }%
7684 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7685 \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
7686 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7687 \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%

```

```

7688 }%
7689 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7690   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7691   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7692   \glsxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7693 }%
7694 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7695   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7696   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7697   \glsxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7698 }%

```

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.

```

7699 \renewcommand*{\glsxtrfullformat}[2]{%
7700   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7701   \ifglxtrinsertinside\else##2\fi
7702 }%
7703 \renewcommand*{\glsxtrfullplformat}[2]{%
7704   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7705   \ifglxtrinsertinside\else##2\fi
7706 }%
7707 \renewcommand*{\Glsxtrfullformat}[2]{%
7708   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7709   \ifglxtrinsertinside\else##2\fi
7710 }%
7711 \renewcommand*{\Glsxtrfullplformat}[2]{%
7712   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7713   \ifglxtrinsertinside\else##2\fi
7714 }%
7715 }

```

ort-nolong-desc

```

7716 \letabbreviationstyle{short-nolong-desc}{short-desc}

```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

7717 \newabbreviationstyle{short-nolong-desc-noreg}%
7718 {%
7719   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

```

Unset the regular attribute if it has been set.

```

7720 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7721   \glshasattribute{\the\glslabeltok}{regular}%
7722   {%
7723     \glssetattribute{\the\glslabeltok}{regular}{false}%
7724   }%
7725   {}%
7726 }%
7727 }%
7728 {%

```

```

7729 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
7730 }

```

**nolong-short** Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

7731 \newabbreviationstyle{nolong-short}%
7732 {%
7733 \GlsXtrUseAbbrStyleSetup{short-nolong}%
7734 }%
7735 {%
7736 \GlsXtrUseAbbrStyleFmts{short-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

7737 \renewcommand*{\glxtrinlinefullformat}[2]{%
7738 \protect\glfirstlongfont{\glsaccesslong{##1}}%
7739 \ifglxtrininsertinside##2\fi}%
7740 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7741 \glxtrparen{\glfirstabbrvfont{\glsaccessshort{##1}}}%
7742 }%
7743 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7744 \protect\glfirstlongfont{\glsaccesslongpl{##1}}%
7745 \ifglxtrininsertinside##2\fi}%
7746 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7747 \glxtrparen{\glfirstabbrvfont{\glsaccessshortpl{##1}}}%
7748 }%
7749 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7750 \protect\glfirstlongfont{\glsaccesslong{##1}}%
7751 \ifglxtrininsertinside##2\fi}%
7752 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7753 \glxtrparen{\glfirstabbrvfont{\Glsaccessshort{##1}}}%
7754 }%
7755 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7756 \protect\glfirstlongfont{\glsaccesslongpl{##1}}%
7757 \ifglxtrininsertinside##2\fi}%
7758 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7759 \glxtrparen{\glfirstabbrvfont{\Glsaccessshortpl{##1}}}%
7760 }%
7761 }

```

**ong-short-noreg** Like nolong-short but doesn't set the regular attribute.

```

7762 \newabbreviationstyle{nolong-short-noreg}%
7763 {%
7764 \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

7765 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7766 \glshasattribute{\the\glslabeltok}{regular}%
7767 {%
7768 \glssetattribute{\the\glslabeltok}{regular}{false}%
7769 }%

```

```

7770    {}%
7771  }%
7772}%
7773{%
7774  \GlsXtrUseAbbrStyleFmts{nolong-short}%
7775}

```

noshortdescname

```

7776 \newcommand*{\glxtrlongnoshortdescname}{%
7777   \protect\glslongfont{\the\glslongtok}%
7778}

```

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.

```

7779 \newabbreviationstyle{long-desc}%
7780 {%
7781   \renewcommand*{\CustomAbbreviationFields}{%
7782     name={\glxtrlongnoshortdescname},
7783     sort={\the\glslongtok},
7784     first={\protect\glslongfont{\the\glslongtok}},
7785     firstplural={\protect\glslongfont{\the\glslongpltok}},
7786     text={\glslongfont{\the\glslongtok}},
7787     plural={\glslongfont{\the\glslongpltok}}}%
7788   }%
7789   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7790     \glsssetattribute{\the\glslabeltok}{regular}{true}}%
7791 }%
7792 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7793 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7794 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvdefaultfont{##1}}%
7795 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
7796 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
7797 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

7798 \renewcommand*{\glxtrsubsequentfmt}[2]{%
7799   \glslongfont{\glssaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7800   \ifglxtrinsertinside \else##2\fi
7801 }%
7802 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
7803   \glslongfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7804   \ifglxtrinsertinside \else##2\fi
7805 }%
7806 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
7807   \glslongfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7808   \ifglxtrinsertinside \else##2\fi
7809 }%

```

```

7810 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
7811   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
7812   \ifglsxtrinsertinside \else##2\fi
7813 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

7814 \renewcommand*{\glsxtrinlinefullformat}[2]{%
7815   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7816   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7817   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7818 }%
7819 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
7820   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7821   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7822   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7823 }%
7824 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7825   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7826   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7827   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7828 }%
7829 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7830   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7831   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7832   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7833 }%

```

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.

```

7834 \renewcommand*{\glsxtrfullformat}[2]{%
7835   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7836   \ifglsxtrinsertinside\else##2\fi
7837 }%
7838 \renewcommand*{\glsxtrfullplformat}[2]{%
7839   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7840   \ifglsxtrinsertinside\else##2\fi
7841 }%
7842 \renewcommand*{\Glsxtrfullformat}[2]{%
7843   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
7844   \ifglsxtrinsertinside\else##2\fi
7845 }%
7846 \renewcommand*{\Glsxtrfullplformat}[2]{%
7847   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
7848   \ifglsxtrinsertinside\else##2\fi
7849 }%
7850 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```

7851 \letabbreviationstyle{long-noshort-desc}{long-desc}

```



short-desc-noreg Like long-noshort-desc but doesn't set the regular attribute.

```
7852 \newabbreviationstyle{long-noshort-desc-noreg}%
7853 {%
7854   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
7855   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7856     \glshasattribute{\the\glslabeltok}{regular}%
7857     {%
7858       \glissetattribute{\the\glslabeltok}{regular}{false}%
7859     }%
7860   }%
7861 }%
7862 }%
7863 {%
7864   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
7865 }
```

longnoshortname

```
7866 \newcommand*{\glsxtrlongnoshortname}{%
7867   \protect\glsabbrvfont{\the\glsshorttok}%
7868 }
```

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.

```
7869 \newabbreviationstyle{long}%
7870 {%
7871   \renewcommand*{\CustomAbbreviationFields}{%
7872     name={\glsxtrlongnoshortname},
7873     sort={\the\glsshorttok},
7874     first={\protect\glsfirstlongfont{\the\glslongtok}},
7875     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7876     text={\glslongfont{\the\glslongtok}},
7877     plural={\glslongfont{\the\glslongpltok}},%
7878     description={\the\glslongtok}%
7879   }%
7880   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7881     \glissetattribute{\the\glslabeltok}{regular}{true}}%
7882 }%
7883 {%
7884   \GlsXtrUseAbbrStyleFmts{long-desc}%
7885 }
```

long-noshort Provide a synonym that matches similar styles.

```
7886 \letabbreviationstyle{long-noshort}{long}
```

g-noshort-noreg Like long-noshort but doesn't set the regular attribute.

```

7887 \newabbreviationstyle{long-noshort-noreg}%
7888 {%
7889   \GlsXtrUseAbbrStyleSetup{long-noshort}%
7890   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7891     \glshasattribute{\the\glslabeltok}{regular}%
7892     {%
7893       \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7894     }%
7895     {}%
7896   }%
7897 }%
7898 {%
7899   \GlsXtrUseAbbrStyleFmts{long-noshort}%
7900 }

```

### 1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

```

\glsxtrscfont   Maintained for backward-compatibility.
7901 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}

\glssabrvscfont Added for consistent naming.
7902 \newcommand*{\glssabrvscfont}{\glsxtrscfont}

sxtrfirstscfont Maintained for backward-compatibility.
7903 \newcommand*{\glsxtrfirstscfont}[1]{\glssabrvscfont{#1}}

irstabrvscfont  Added for consistent naming.
7904 \newcommand*{\glsfirstabrvscfont}{\glsxtrfirstscfont}

```

and for the default short form suffix:

```

\glsxtrscsuffix
7905 \newcommand*{\glsxtrscsuffix}{\glstextup{\glxtrabrvpluralsuffix}}

long-short-sc
7906 \newabbreviationstyle{long-short-sc}%
7907 {%
7908   \renewcommand*{\CustomAbbreviationFields}{%
7909     name={\glsxtrlongshortname},
7910     sort={\the\glsshorttok},
7911     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7912       \protect\glsxtrfullsep{\the\glslabeltok}%
7913       \glsxtrparen{\protect\glsfirstabrvscfont{\the\glsshorttok}}},%
7914     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7915       \protect\glsxtrfullsep{\the\glslabeltok}%

```

```

7916 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7917 plural={\protect\glsabbrvscfont{\the\glsshortpltok}},%
7918 description={\the\glslongtok}}%
7919 \renewcommand*\GlsXtrPostNewAbbreviation}{%
7920 \glsattribute{\the\glslabeltok}{regular}%
7921 {%
7922 \glssetattribute{\the\glslabeltok}{regular}{false}%
7923 }%
7924 {}%
7925 }%
7926 }%
7927 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7928 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
7929 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
7930 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%

```

Use the default long fonts.

```

7931 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7932 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7933 \renewcommand*\glxtrfullformat[2]{%
7934 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7935 \ifglxtrinsertinside\else##2\fi
7936 \glxtrfullsep{##1}}%
7937 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7938 }%
7939 \renewcommand*\glxtrfullplformat[2]{%
7940 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7941 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}}%
7942 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7943 }%
7944 \renewcommand*\Glsxtrfullformat[2]{%
7945 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7946 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}}%
7947 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7948 }%
7949 \renewcommand*\Glsxtrfullplformat[2]{%
7950 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7951 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}}%
7952 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7953 }%
7954 }

```

g-short-sc-desc

```

7955 \newabbreviationstyle{long-short-sc-desc}%
7956 {%
7957 \renewcommand*\CustomAbbreviationFields}{%

```

```

7958   name={\glxtrlongshortdescname},
7959   sort={\glxtrlongshortdescsort},%
7960   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7961     \protect\glxtrfullsep{\the\glslabeltok}%
7962     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7963   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7964     \protect\glxtrfullsep{\the\glslabeltok}%
7965     \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7966   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7967   plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7968 }%

```

Unset the regular attribute if it has been set.

```

7969 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7970   \glshasattribute{\the\glslabeltok}{regular}%
7971   {%
7972     \glissetattribute{\the\glslabeltok}{regular}{false}%
7973   }%
7974   {}%
7975 }%
7976 }%
7977 {%

```

As long-short-sc style:

```

7978 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
7979 }

```

Now the short (long) version

```

7980 \newabbreviationstyle{short-sc-long}%
7981 {%
7982   \renewcommand*{\CustomAbbreviationFields}{%
7983     name={\glxtrshortlongname},
7984     sort={\the\glsshorttok},
7985     description={\the\glslongtok},%
7986     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
7987       \protect\glxtrfullsep{\the\glslabeltok}%
7988       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
7989     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7990       \protect\glxtrfullsep{\the\glslabeltok}%
7991       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
7992     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

7993 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7994   \glshasattribute{\the\glslabeltok}{regular}%
7995   {%
7996     \glissetattribute{\the\glslabeltok}{regular}{false}%
7997   }%
7998   {}%
7999 }%
8000 }%

```

8001 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8002 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8003 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8004 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8005 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8006 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8007 \renewcommand*{\glxtrfullformat}[2]{%
8008   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8009   \ifglxtrininsertinside\else##2\fi
8010   \glxtrfullsep{##1}%
8011   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8012 }%
8013 \renewcommand*{\glxtrfullplformat}[2]{%
8014   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8015   \ifglxtrininsertinside\else##2\fi
8016   \glxtrfullsep{##1}%
8017   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8018 }%
8019 \renewcommand*{\Glsxtrfullformat}[2]{%
8020   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8021   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8022   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8023 }%
8024 \renewcommand*{\Glsxtrfullplformat}[2]{%
8025   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8026   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8027   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8028 }%
8029 }
```

As before but user provides description

```
8030 \newabbreviationstyle{short-sc-long-desc}%
8031 {%
8032   \renewcommand*{\CustomAbbreviationFields}{%
8033     name={\glxtrshortlongdescname},
8034     sort={\glxtrshortlongdescsort},
8035     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}}%
8036     \protect\glxtrfullsep{\the\glslabeltok}%
8037     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8038     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}}%
8039     \protect\glxtrfullsep{\the\glslabeltok}%
8040     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8041     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8042     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
8043   }%
```

Unset the regular attribute if it has been set.

```

8044 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8045   \glshasattribute{\the\glslabeltok}{regular}}%
8046   {%
8047     \glissetattribute{\the\glslabeltok}{regular}{false}}%
8048   }%
8049   {}%
8050 }%
8051 }%
8052 {%

```

As short-sc-long style:

```

8053 \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8054 }

```

short-sc

```

8055 \newabbreviationstyle{short-sc}%
8056 {%
8057   \renewcommand*{\CustomAbbreviationFields}{%
8058     name={\glxtrshortnolongname},
8059     sort={\the\glsshorttok},
8060     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8061     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8062     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8063     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8064     description={\the\glslongtok}}%
8065   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8066     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8067   }%
8068 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8069 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8070 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8071 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8072 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8073 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8074 \renewcommand*{\glxtrinlinefullformat}[2]{%
8075   \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
8076   \ifglxtrininsertinside##2\fi}%
8077   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8078   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8079 }%
8080 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8081   \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
8082   \ifglxtrininsertinside##2\fi}%
8083   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8084   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8085 }%

```

```

8086 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8087   \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}%
8088   \ifglsxtrininsertinside##2\fi}%
8089   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8090   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8091 }%
8092 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8093   \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}%
8094   \ifglsxtrininsertinside##2\fi}%
8095   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8096   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8097 }%

```

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.

```

8098 \renewcommand*{\glsxtrfullformat}[2]{%
8099   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8100   \ifglsxtrininsertinside\else##2\fi
8101 }%
8102 \renewcommand*{\glsxtrfullplformat}[2]{%
8103   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8104   \ifglsxtrininsertinside\else##2\fi
8105 }%
8106 \renewcommand*{\Glsxtrfullformat}[2]{%
8107   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8108   \ifglsxtrininsertinside\else##2\fi
8109 }%
8110 \renewcommand*{\Glsxtrfullplformat}[2]{%
8111   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8112   \ifglsxtrininsertinside\else##2\fi
8113 }%
8114 }

```

short-sc-nolong

```

8115 \letabbreviationstyle{short-sc-nolong}{short-sc}

```

short-sc-desc

```

8116 \newabbreviationstyle{short-sc-desc}%
8117 {%
8118   \renewcommand*{\CustomAbbreviationFields}{%
8119     name={\glsxtrshortdesname},
8120     sort={\the\glsshorttok},
8121     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8122     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8123     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8124     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8125     description={\the\glslongtok}}%
8126 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8127   \glssetattribute{\the\glslabeltok}{regular}{true}}%

```

8128 }%  
 8129 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
8130 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
8131 \renewcommand*\glssabrvfont[1]{\glssabrvscfont{##1}}%
8132 \renewcommand*\glsfirstabrvfont[1]{\glsfirstabrvscfont{##1}}%
8133 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8134 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
8135 \renewcommand*\glxtrinlinefullformat[2]{%
8136   \glsfirstabrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8137   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8138   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8139 }%
8140 \renewcommand*\glxtrinlinefullplformat[2]{%
8141   \glsfirstabrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8142   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8143   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8144 }%
8145 \renewcommand*\Glsxtrinlinefullformat[2]{%
8146   \glsfirstabrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8147   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8148   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8149 }%
8150 \renewcommand*\Glsxtrinlinefullplformat[2]{%
8151   \glsfirstabrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8152   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8153   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8154 }%
```

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.

```
8155 \renewcommand*\glxtrfullformat[2]{%
8156   \glsfirstabrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8157   \ifglxtrininsertinside\else##2\fi
8158 }%
8159 \renewcommand*\glxtrfullplformat[2]{%
8160   \glsfirstabrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8161   \ifglxtrininsertinside\else##2\fi
8162 }%
8163 \renewcommand*\Glsxtrfullformat[2]{%
8164   \glsfirstabrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8165   \ifglxtrininsertinside\else##2\fi
8166 }%
8167 \renewcommand*\Glsxtrfullplformat[2]{%
8168   \glsfirstabrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8169   \ifglxtrininsertinside\else##2\fi
8170 }%
8171 }
```



-sc-nolong-desc

```
8172 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

nolong-short-sc

```
8173 \newabbreviationstyle{nolong-short-sc}%
8174 {%
8175   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8176 }%
8177 {%
8178   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
8179   \renewcommand*{\glxtrinlinefullformat}[2]{%
8180     \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8181       \ifglxtrininsertinside##2\fi}%
8182     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8183     \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8184   }%
8185   \renewcommand*{\glxtrinlinefullplformat}[2]{%
8186     \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8187       \ifglxtrininsertinside##2\fi}%
8188     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8189     \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8190   }%
8191   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8192     \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8193       \ifglxtrininsertinside##2\fi}%
8194     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8195     \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8196   }%
8197   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8198     \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8199       \ifglxtrininsertinside##2\fi}%
8200     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8201     \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8202   }%
8203 }
```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```
8204 \newabbreviationstyle{long-noshort-sc}%
8205 {%
8206   \renewcommand*{\CustomAbbreviationFields}{%
8207     name={\glxtrlongnoshortname},
8208     sort={\the\glsshorttok},
8209     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8210     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8211     text={\protect\glslongdefaultfont{\the\glslongtok}},
8212     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
```

```

8213   description={\the\glslongtok}%
8214 }%
8215 \renewcommand*\GlsXtrPostNewAbbreviation}{%
8216   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8217 }%
8218 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8219 \renewcommand*\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
8220 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8221 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8222 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8223 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8224 \renewcommand*\glsxtrsubsequentfmt}[2]{%
8225   \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8226   \ifglsxtrininsertinside \else##2\fi
8227 }%
8228 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
8229   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8230   \ifglsxtrininsertinside \else##2\fi
8231 }%
8232 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
8233   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8234   \ifglsxtrininsertinside \else##2\fi
8235 }%
8236 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
8237   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8238   \ifglsxtrininsertinside \else##2\fi
8239 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8240 \renewcommand*\glsxtrinlinefullformat}[2]{%
8241   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8242   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8243   \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8244 }%
8245 \renewcommand*\glsxtrinlinefullplformat}[2]{%
8246   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8247   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8248   \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8249 }%
8250 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8251   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8252   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8253   \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8254 }%
8255 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8256   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8257   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```

```

8258 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8259 }%

```

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.

```

8260 \renewcommand*{\glxtrfullformat}[2]{%
8261 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8262 \ifglxtrininsertinside\else##2\fi
8263 }%
8264 \renewcommand*{\glxtrfullplformat}[2]{%
8265 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8266 \ifglxtrininsertinside\else##2\fi
8267 }%
8268 \renewcommand*{\Glsxtrfullformat}[2]{%
8269 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8270 \ifglxtrininsertinside\else##2\fi
8271 }%
8272 \renewcommand*{\Glsxtrfullplformat}[2]{%
8273 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8274 \ifglxtrininsertinside\else##2\fi
8275 }%
8276 }

```

long-sc Backward compatibility:

```

8277 \@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`.

```

8278 \newabbreviationstyle{long-noshort-sc-desc}%
8279 {%
8280 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8281 }%
8282 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8283 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8284 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8285 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8286 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8287 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8288 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8289 \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
8290 \ifglxtrininsertinside \else##2\fi
8291 }%
8292 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8293 \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
8294 \ifglxtrininsertinside \else##2\fi
8295 }%

```

```

8296 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8297   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8298   \ifglxtrinsertinside \else##2\fi
8299 }%
8300 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8301   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8302   \ifglxtrinsertinside \else##2\fi
8303 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8304 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8305   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8306   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8307   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8308 }%
8309 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8310   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8311   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8312   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8313 }%
8314 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8315   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8316   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8317   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8318 }%
8319 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8320   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8321   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8322   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8323 }%

```

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.

```

8324 \renewcommand*{\glxtrfullformat}[2]{%
8325   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8326   \ifglxtrinsertinside\else##2\fi
8327 }%
8328 \renewcommand*{\glxtrfullplformat}[2]{%
8329   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8330   \ifglxtrinsertinside\else##2\fi
8331 }%
8332 \renewcommand*{\Glsxtrfullformat}[2]{%
8333   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8334   \ifglxtrinsertinside\else##2\fi
8335 }%
8336 \renewcommand*{\Glsxtrfullplformat}[2]{%
8337   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8338   \ifglxtrinsertinside\else##2\fi
8339 }%
8340 }

```

long-desc-sc Backward compatibility:

```
8341 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
8342 \newabbreviationstyle{short-sc-footnote}%
8343 {%
8344   \renewcommand*{\CustomAbbreviationFields}{%
8345     name={\glxtrfootnotename},
8346     sort={\the\glsshorttok},
8347     description={\the\glslongtok},%
8348     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8349       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8350       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8351     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8352       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8353       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8354     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8355   }
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
8355   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8356     \glsssetAttribute{\the\glslabeltok}{nohyperfirst}{true}%
8357     \glshasattribute{\the\glslabeltok}{regular}%
8358     {%
8359       \glsssetAttribute{\the\glslabeltok}{regular}{false}%
8360     }%
8361   }%
8362 }%
8363 }%
8364 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8365 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8366 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8367 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8368 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8369 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
8370 \renewcommand*{\glxtrfullformat}[2]{%
8371   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8372   \ifglxtrininsertinside\else##2\fi
8373   \protect\glxtrabbrvfootnote{##1}%
8374   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8375 }%
8376 \renewcommand*{\glxtrfullplformat}[2]{%
8377   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8378   \ifglxtrininsertinside\else##2\fi
8379   \protect\glxtrabbrvfootnote{##1}%
8380   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8381 }
```

```

8381 }%
8382 \renewcommand*{\Glsxtrfullformat}[2]{%
8383   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8384   \ifglxtrinsertinside\else##2\fi
8385   \protect\glxtrabbrvfootnote{##1}%
8386   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8387 }%
8388 \renewcommand*{\Glsxtrfullplformat}[2]{%
8389   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8390   \ifglxtrinsertinside\else##2\fi
8391   \protect\glxtrabbrvfootnote{##1}%
8392   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8393 }%

```

The first use full form and the inline full form use the short (long) style.

```

8394 \renewcommand*{\glxtrinlinefullformat}[2]{%
8395   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8396   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8397   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8398 }%
8399 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8400   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8401   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8402   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8403 }%
8404 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8405   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8406   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8407   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8408 }%
8409 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8410   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8411   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8412   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8413 }%
8414 }

```

footnote-sc Backward compatibility:

```

8415 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

8416 \newabbreviationstyle{short-sc-postfootnote}%
8417 {%
8418   \renewcommand*{\CustomAbbreviationFields}{%
8419     name={\glxtrfootnotename},
8420     sort={\the\glsshorttok},
8421     description={\the\glslongtok},%
8422     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
8423     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
8424     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.

```
8425 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8426 \csdef{glxstrpostlink\glscategorylabel}{%
8427 \glxtrifwasfirstuse
8428 }
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
8429 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8430 {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
8431 }%
8432 {}%
8433 }%
8434 \glshasattribute{\the\glslabeltok}{regular}%
8435 {%
8436 \glissetattribute{\the\glslabeltok}{regular}{false}%
8437 }%
8438 {}%
8439 }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
8440 \renewcommand*{\glxtrsetupfulldefs}{%
8441 \let\glxtrifwasfirstuse\@secondoftwo
8442 }%
8443 }%
8444 }
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8445 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8446 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
8447 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8448 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8449 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
8450 \renewcommand*{\glxtrfullformat}[2]{%
8451 \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8452 \ifglxtrininsertinside\else##2\fi
8453 }%
8454 \renewcommand*{\glxtrfullplformat}[2]{%
8455 \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8456 \ifglxtrininsertinside\else##2\fi
8457 }%
8458 \renewcommand*{\Glsxtrfullformat}[2]{%
8459 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8460 \ifglxtrininsertinside\else##2\fi
8461 }%
8462 \renewcommand*{\Glsxtrfullplformat}[2]{%

```

```

8463 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8464 \ifglxtrinsertinside\else##2\fi
8465 }%

```

The first use full form and the inline full form use the short (long) style.

```

8466 \renewcommand*{\glxtrinlinefullformat}[2]{%
8467 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8468 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8469 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8470 }%
8471 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8472 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8473 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8474 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8475 }%
8476 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8477 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8478 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8479 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8480 }%
8481 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8482 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8483 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8484 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8485 }%
8486 }

```

postfootnote-sc Backward compatibility:

```

8487 \@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.
8488 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

\glsabbrvsmfont Added for consistent naming.
8489 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}

sxtrfirstsmfont Maintained for backward compatibility.
8490 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}

irstabbrvsmfont Added for consistent naming.
8491 \newcommand*{\glsfirstabbrvsmfont}{\glxtrfirstsmfont}

```

and for the default short form suffix:



\glxtrsmsuffix

```
8492 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}
```

long-short-sm

```
8493 \newabbreviationstyle{long-short-sm}%
8494 {%
8495   \renewcommand*{\CustomAbbreviationFields}{%
8496     name={\glxtrlongshortname},
8497     sort={\the\glsshorttok},
8498     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8499       \protect\glxtrfullsep{\the\glslabeltok}%
8500       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8501     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8502       \protect\glxtrfullsep{\the\glslabeltok}%
8503       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8504     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}},%
8505     description={\the\glslongtok}}%
8506   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8507     \glshasattribute{\the\glslabeltok}{regular}%
8508     {%
8509       \glissetattribute{\the\glslabeltok}{regular}{false}%
8510     }%
8511     {}%
8512   }%
8513 }%
8514 {%
8515   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8516   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8517   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%

```

Use the default long fonts.

```
8518 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8519 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8520 \renewcommand*{\glxtrfullformat}[2]{%
8521   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8522   \ifglxtrininsertinside\else##2\fi
8523   \glxtrfullsep{##1}%
8524   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8525 }%
8526 \renewcommand*{\glxtrfullplformat}[2]{%
8527   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8528   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8529   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8530 }%
8531 \renewcommand*{\Glsxtrfullformat}[2]{%
8532   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8533   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8534   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%

```

```

8535 }%
8536 \renewcommand*{\Glsxtrfullplformat}[2]{%
8537   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8538   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8539   \glsxtrparen{\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8540 }%
8541 }

```

#### g-short-sm-desc

```

8542 \newabbreviationstyle{long-short-sm-desc}%
8543 {%
8544   \renewcommand*{\CustomAbbreviationFields}{%
8545     name={\glsxtrlongshortdescname},
8546     sort={\glsxtrlongshortdescsort},%
8547     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8548       \protect\glsxtrfullsep{\the\glslabeltok}%
8549       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8550     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8551       \protect\glsxtrfullsep{\the\glslabeltok}%
8552       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8553     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8554     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8555   }%

```

Unset the regular attribute if it has been set.

```

8556 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8557   \glshasattribute{\the\glslabeltok}{regular}%
8558   {%
8559     \glissetattribute{\the\glslabeltok}{regular}{false}%
8560   }%
8561   {}%
8562 }%
8563 }%
8564 {%

```

As long-short-sm style:

```

8565   \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8566 }

```

#### short-sm-long Now the short (long) version

```

8567 \newabbreviationstyle{short-sm-long}%
8568 {%
8569   \renewcommand*{\CustomAbbreviationFields}{%
8570     name={\glsxtrshortlongname},
8571     sort={\the\glsshorttok},
8572     description={\the\glslongtok},%
8573     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8574       \protect\glsxtrfullsep{\the\glslabeltok}%
8575       \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8576     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%

```

```

8577 \protect\glxtrfullsep{\the\glslabeltok}%
8578 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8579 plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8580 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8581 \glshasattribute{\the\glslabeltok}{regular}%
8582 {%
8583 \glissetattribute{\the\glslabeltok}{regular}{false}%
8584 }%
8585 {}%
8586 }%
8587 }%
8588 {%
8589 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8590 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8591 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8592 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8593 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8594 \renewcommand*{\glxtrfullformat}[2]{%
8595 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8596 \ifglxtrininsertinside\else##2\fi
8597 \glxtrfullsep{##1}%
8598 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8599 }%
8600 \renewcommand*{\glxtrfullplformat}[2]{%
8601 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8602 \ifglxtrininsertinside\else##2\fi
8603 \glxtrfullsep{##1}%
8604 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8605 }%
8606 \renewcommand*{\Glsxtrfullformat}[2]{%
8607 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8608 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8609 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8610 }%
8611 \renewcommand*{\Glsxtrfullplformat}[2]{%
8612 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8613 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8614 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8615 }%
8616 }

```

rt-sm-long-desc As before but user provides description

```

8617 \newabbreviationstyle{short-sm-long-desc}%
8618 {%
8619 \renewcommand*{\CustomAbbreviationFields}{%
8620 name={\glxtrshortlongdescname},

```

```

8621     sort={\glxtrshortlongdescsort},
8622     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8623       \protect\glxtrfullsep{\the\glslabeltok}%
8624       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8625     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8626       \protect\glxtrfullsep{\the\glslabeltok}%
8627       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8628     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8629     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8630 }%

```

Unset the regular attribute if it has been set.

```

8631 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8632   \glshasattribute{\the\glslabeltok}{regular}%
8633   {%
8634     \glissetattribute{\the\glslabeltok}{regular}{false}%
8635   }%
8636   {}%
8637 }%
8638 }%
8639 {%

```

As short-sm-long style:

```

8640 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8641 }

```

short-sm

```

8642 \newabbreviationstyle{short-sm}%
8643 {%
8644   \renewcommand*{\CustomAbbreviationFields}{%
8645     name={\glxtrshortnolongname},
8646     sort={\the\glsshorttok},
8647     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8648     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8649     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8650     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8651     description={\the\glslongtok}}%
8652   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8653     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8654 }%
8655 {%
8656   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8657   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8658   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8659   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8660   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8661 \renewcommand*{\glxtrinlinefullformat}[2]{%
8662   \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}%

```

```

8663     \ifglxtrinsertinside##2\fi}%
8664     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8665     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8666 }%
8667 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8668     \protect\glsfirstabbrvsmfont{\glssaccessshortpl{##1}%
8669     \ifglxtrinsertinside##2\fi}%
8670     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8671     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8672 }%

8673 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8674     \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}%
8675     \ifglxtrinsertinside##2\fi}%
8676     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8677     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8678 }%
8679 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8680     \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}%
8681     \ifglxtrinsertinside##2\fi}%
8682     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8683     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8684 }%

```

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.

```

8685 \renewcommand*{\glxtrfullformat}[2]{%
8686     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8687     \ifglxtrinsertinside\else##2\fi
8688 }%
8689 \renewcommand*{\glxtrfullplformat}[2]{%
8690     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8691     \ifglxtrinsertinside\else##2\fi
8692 }%
8693 \renewcommand*{\Glsxtrfullformat}[2]{%
8694     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8695     \ifglxtrinsertinside\else##2\fi
8696 }%
8697 \renewcommand*{\Glsxtrfullplformat}[2]{%
8698     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8699     \ifglxtrinsertinside\else##2\fi
8700 }%
8701 }

```

short-sm-nolong

```
8702 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
8703 \newabbreviationstyle{short-sm-desc}%

```

```

8704 {%
8705   \renewcommand*{\CustomAbbreviationFields}{%
8706     name={\glxtrshortdescname},
8707     sort={\the\glsshorttok},
8708     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8709     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8710     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8711     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8712     description={\the\glslongtok}}%
8713   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8714     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8715 }%
8716 {%
8717   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8718   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8719   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8720   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8721   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8722   \renewcommand*{\glxtrinlinefullformat}[2]{%
8723     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8724     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8725   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8726 }%
8727   \renewcommand*{\glxtrinlinefullplformat}[2]{%
8728     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8729     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8730   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8731 }%
8732   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8733     \glsfirstabbrvsmfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8734     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8735   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8736 }%
8737   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8738     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8739     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8740   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8741 }%

```

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.

```

8742   \renewcommand*{\glxtrfullformat}[2]{%
8743     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8744     \ifglxtrininsertinside\else##2\fi
8745 }%
8746   \renewcommand*{\glxtrfullplformat}[2]{%
8747     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8748     \ifglxtrininsertinside\else##2\fi

```

```

8749 }%
8750 \renewcommand*{\Glsxtrfullformat}[2]{%
8751   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8752   \ifglxtrinsertinside\else##2\fi
8753 }%
8754 \renewcommand*{\Glsxtrfullplformat}[2]{%
8755   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8756   \ifglxtrinsertinside\else##2\fi
8757 }%
8758 }

```

-sm-nolong-desc

```

8759 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

8760 \newabbreviationstyle{nolong-short-sm}%
8761 {%
8762   \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
8763 }%
8764 {%
8765   \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8766 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8767   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8768     \ifglxtrinsertinside##2\fi}%
8769   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8770   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8771 }%
8772 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8773   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8774     \ifglxtrinsertinside##2\fi}%
8775   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8776   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8777 }%
8778 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8779   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8780     \ifglxtrinsertinside##2\fi}%
8781   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8782   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8783 }%
8784 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8785   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8786     \ifglxtrinsertinside##2\fi}%
8787   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8788   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8789 }%
8790 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8791 \newabbreviationstyle{long-noshort-sm}%
8792 {%
8793   \renewcommand*{\CustomAbbreviationFields}{%
8794     name={\glsxtrlongnoshortname},
8795     sort={\the\glsshorttok},
8796     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8797     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8798     text={\protect\glslongdefaultfont{\the\glslongtok}},
8799     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8800     description={\the\glslongtok}%
8801   }%
8802   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8803     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8804 }%
8805 {%
8806   \renewcommand*{\glabbrvfont}[1]{\glabbrvsmfont{##1}}%
8807   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8808   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
8809   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8810   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8811 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
8812   \glslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8813   \ifglsxtrininsertinside \else##2\fi
8814 }%
8815 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
8816   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8817   \ifglsxtrininsertinside \else##2\fi
8818 }%
8819 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8820   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8821   \ifglsxtrininsertinside \else##2\fi
8822 }%
8823 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8824   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8825   \ifglsxtrininsertinside \else##2\fi
8826 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8827 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8828   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8829   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8830   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8831 }%
8832 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8833   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8834   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```



```

8835 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8836 }%
8837 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8838 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8839 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8840 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8841 }%
8842 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8843 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8844 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8845 \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8846 }%

```

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.

```

8847 \renewcommand*{\glsxtrfullformat}[2]{%
8848 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8849 \ifglsxtrininsertinside\else##2\fi
8850 }%
8851 \renewcommand*{\glsxtrfullplformat}[2]{%
8852 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8853 \ifglsxtrininsertinside\else##2\fi
8854 }%
8855 \renewcommand*{\Glsxtrfullformat}[2]{%
8856 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8857 \ifglsxtrininsertinside\else##2\fi
8858 }%
8859 \renewcommand*{\Glsxtrfullplformat}[2]{%
8860 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8861 \ifglsxtrininsertinside\else##2\fi
8862 }%
8863 }

```

**long-sm** Backward compatibility:

```

8864 \@glsxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}

```

**noshort-sm-desc** The smaller font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8865 \newabbreviationstyle{long-noshort-sm-desc}%
8866 {%
8867 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8868 }%
8869 {%
8870 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8871 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8872 \renewcommand*\abbrvpluralsuffix{\protect\glsxtrsmsuffix}%
8873 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8874 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8875 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8876   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8877   \ifglxtrinsertinside \else##2\fi
8878 }%
8879 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8880   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8881   \ifglxtrinsertinside \else##2\fi
8882 }%
8883 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8884   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8885   \ifglxtrinsertinside \else##2\fi
8886 }%
8887 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8888   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8889   \ifglxtrinsertinside \else##2\fi
8890 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8891 \renewcommand*{\glxtrinlinefullformat}[2]{%
8892   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8893   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8894   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8895 }%
8896 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8897   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8898   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8899   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8900 }%
8901 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8902   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8903   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8904   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8905 }%
8906 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8907   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8908   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8909   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8910 }%

```

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.

```

8911 \renewcommand*{\glxtrfullformat}[2]{%
8912   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8913   \ifglxtrinsertinside\else##2\fi
8914 }%
8915 \renewcommand*{\glxtrfullplformat}[2]{%
8916   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8917   \ifglxtrinsertinside\else##2\fi
8918 }%
8919 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

8920 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8921 \ifglxtrinsertinside\else##2\fi
8922 }%
8923 \renewcommand*{\Glsxtrfullplformat}[2]{%
8924 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8925 \ifglxtrinsertinside\else##2\fi
8926 }%
8927 }

```

long-desc-sm Backward compatibility:

```

8928 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

8929 \newabbreviationstyle{short-sm-footnote}%
8930 {%
8931 \renewcommand*{\CustomAbbreviationFields}{%
8932 name={\glxtrfootnotename},
8933 sort={\the\glsshorttok},
8934 description={\the\glslongtok},%
8935 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8936 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8937 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8938 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8939 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8940 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8941 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.

```

8942 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8943 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8944 \glsattribute{\the\glslabeltok}{regular}%
8945 {%
8946 \glssetattribute{\the\glslabeltok}{regular}{false}%
8947 }%
8948 {}%
8949 }%
8950 }%
8951 {%
8952 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8953 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8954 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8955 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8956 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8957 \renewcommand*{\glxtrfullformat}[2]{%
8958 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8959 \ifglxtrinsertinside\else##2\fi
8960 \protect\glxtrabbrvfootnote{##1}%

```

```

8961     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8962 }%
8963 \renewcommand*{\glsxtrfullplformat}[2]{%
8964   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8965   \ifglsxtrininsertinside\else##2\fi
8966   \protect\glsxtrabbrvfootnote{##1}%
8967   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8968 }%
8969 \renewcommand*{\Glsxtrfullformat}[2]{%
8970   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8971   \ifglsxtrininsertinside\else##2\fi
8972   \protect\glsxtrabbrvfootnote{##1}%
8973   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8974 }%
8975 \renewcommand*{\Glsxtrfullplformat}[2]{%
8976   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8977   \ifglsxtrininsertinside\else##2\fi
8978   \protect\glsxtrabbrvfootnote{##1}%
8979   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8980 }%

```

The first use full form and the inline full form use the short (long) style.

```

8981 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8982   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8983   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8984   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8985 }%
8986 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8987   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8988   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8989   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8990 }%
8991 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8992   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8993   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8994   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8995 }%
8996 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8997   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8998   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8999   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9000 }%
9001 }

```

footnote-sm Backward compatibility:

```

9002 \@glsxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

9003 \newabbreviationstyle{short-sm-postfootnote}%
9004 {}

```

```

9005 \renewcommand*{\CustomAbbreviationFields}{%
9006   name={\glxtrfootnotename},
9007   sort={\the\glsshorttok},
9008   description={\the\glslongtok},%
9009   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
9010   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
9011   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.

```

9012 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9013   \csdef{glxtrpostlink\glscategorylabel}{%
9014     \glxtrifwasfirstuse
9015     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9016       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}}%
9017       {\glsfirstlongfootnotefont{\glentrylong{\glslabel}}}}%
9018   }%
9019   {%
9020   }%
9021   \glshasattribute{\the\glslabeltok}{regular}%
9022   {%
9023     \glssetattribute{\the\glslabeltok}{regular}{false}%
9024   }%
9025   {%
9026   }%

```

The footnote needs to be suppressed in the inline form, so \glxtrfull must set the first use switch off.

```

9027 \renewcommand*{\glxtrsetupfulldefs}{%
9028   \let\glxtrifwasfirstuse\@secondoftwo
9029 }%
9030}%
9031{%
9032 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9033 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9034 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
9035 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9036 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9037 \renewcommand*{\glxtrfullformat}[2]{%
9038   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9039   \ifglxtrininsertinside\else##2\fi
9040 }%
9041 \renewcommand*{\glxtrfullplformat}[2]{%
9042   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9043   \ifglxtrininsertinside\else##2\fi
9044 }%

```

```

9045 \renewcommand*{\Glsxtrfullformat}[2]{%
9046   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9047   \ifglxtrinsertinside\else##2\fi
9048 }%
9049 \renewcommand*{\Glsxtrfullplformat}[2]{%
9050   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9051   \ifglxtrinsertinside\else##2\fi
9052 }%

```

The first use full form and the inline full form use the short (long) style.

```

9053 \renewcommand*{\glxtrinlinefullformat}[2]{%
9054   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9055   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9056   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9057 }%
9058 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9059   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9060   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9061   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9062 }%
9063 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9064   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9065   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9066   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9067 }%
9068 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9069   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9070   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9071   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9072 }%
9073 }

```

postfootnote-sm Backward compatibility:

```

9074 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

## 1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

9075 \newcommand*{\glsabbrvemfont}[1]{\emph{##1}}%

```

`\glsfirstabbrvemfont`

```

9076 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{##1}}%

```

The default short form suffix:

`\glxtremsuffix`

```

9077 \newcommand*{\glxtremsuffix}{\glxtrabbrvpluralsuffix}

```

`firstlongemfont` Only used by the “long-em” styles.

```
9078 \newcommand*{\glfirstlongemfont}[1]{\glslongemfont{#1}}%
```

`\glslongemfont` Only used by the “long-em” styles.

```
9079 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`long-short-em` The long form is just set in the default long font.

```
9080 \newabbreviationstyle{long-short-em}%
9081 {%
9082   \renewcommand*{\CustomAbbreviationFields}{%
9083     name={\glsxtrlongshortname},
9084     sort={\the\glsshorttok},
9085     first={\protect\glfirstlongdefaultfont{\the\glslongtok}%
9086       \protect\glsxtrfullsep{\the\glslabeltok}%
9087       \glsxtrparen{\protect\glfirstabbrvemfont{\the\glsshorttok}}},%
9088     firstplural={\protect\glfirstlongdefaultfont{\the\glslongpltok}%
9089       \protect\glsxtrfullsep{\the\glslabeltok}%
9090       \glsxtrparen{\protect\glfirstabbrvemfont{\the\glsshortpltok}}},%
9091     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
9092     description={\the\glslongtok}}%
9093   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9094     \glshasattribute{\the\glslabeltok}{regular}}%
9095   {%
9096     \glissetattribute{\the\glslabeltok}{regular}{false}%
9097   }%
9098   {}%
9099 }%
9100 }%
9101 {%
9102   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9103   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9104   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrremsuffix}%

```

Use the default long fonts.

```
9105 \renewcommand*{\glfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9106 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9107 \renewcommand*{\glsxtrfullformat}[2]{%
9108   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9109   \ifglsxtrininsertinside\else##2\fi
9110   \glsxtrfullsep{##1}%
9111   \glsxtrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
9112 }%
9113 \renewcommand*{\glsxtrfullplformat}[2]{%
9114   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9115   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9116   \glsxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9117 }%
9118 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9119 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9120 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9121 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9122 }%
9123 \renewcommand*{\Glsxtrfullplformat}[2]{%
9124 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9125 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9126 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9127 }%
9128 }

```

g-short-em-desc

```

9129 \newabbreviationstyle{long-short-em-desc}%
9130 {%
9131 \renewcommand*{\CustomAbbreviationFields}{%
9132 name={\glxtrlongshortdescname},
9133 sort={\glxtrlongshortdescsort},%
9134 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9135 \protect\glxtrfullsep{\the\glslabeltok}%
9136 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9137 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9138 \protect\glxtrfullsep{\the\glslabeltok}%
9139 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9140 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9141 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9142 }%

```

Unset the regular attribute if it has been set.

```

9143 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9144 \glshasattribute{\the\glslabeltok}{regular}%
9145 {%
9146 \glissetattribute{\the\glslabeltok}{regular}{false}%
9147 }%
9148 {}%
9149 }%
9150 }%
9151 {%

```

As long-short-em style:

```

9152 \GlsXtrUseAbbrStyleFmts{long-short-em}%
9153 }

```

ong-em-short-em

```

9154 \newabbreviationstyle{long-em-short-em}%
9155 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9156 \renewcommand*{\CustomAbbreviationFields}{%
9157 name={\glxtrlongshortname},
9158 sort={\the\glsshorttok},

```



```

9159 first={\protect\glsfirstlongemfont{\the\glslongtok}%
9160 \protect\glsxtrfullsep{\the\glslabeltok}%
9161 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9162 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9163 \protect\glsxtrfullsep{\the\glslabeltok}%
9164 \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

9165 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
9166 description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9167 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9168 \glsattribute{\the\glslabeltok}{regular}%
9169 {%
9170 \glssetattribute{\the\glslabeltok}{regular}{false}%
9171 }%
9172 {}%
9173 }%
9174 }%
9175 {%
9176 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9177 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9178 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9179 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9180 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9181 \renewcommand*{\glsxtrfullformat}[2]{%
9182 \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9183 \ifglsxtrininsertinside\else##2\fi
9184 \glsxtrfullsep{##1}%
9185 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9186 }%
9187 \renewcommand*{\glsxtrfullplformat}[2]{%
9188 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9189 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9190 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9191 }%
9192 \renewcommand*{\Glsxtrfullformat}[2]{%
9193 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9194 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9195 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9196 }%
9197 \renewcommand*{\Glsxtrfullplformat}[2]{%
9198 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9199 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9200 \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9201 }%
9202 }

```

m-short-em-desc

```
9203 \newabbreviationstyle{long-em-short-em-desc}%
9204 {%
9205   \renewcommand*{\CustomAbbreviationFields}{%
9206     name={\glxtrlongshortdescname},
9207     sort={\glxtrlongshortdescsort},%
9208     first={\protect\glsfirstlongemfont{\the\glslongtok}%
9209       \protect\glxtrfullsep{\the\glslabeltok}%
9210       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
9211     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
9212       \protect\glxtrfullsep{\the\glslabeltok}%
9213       \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
9214     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
9215     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9216   }%
```

Unset the regular attribute if it has been set.

```
9217   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9218     \glshasattribute{\the\glslabeltok}{regular}%
9219     {%
9220       \glissetattribute{\the\glslabeltok}{regular}{false}%
9221     }%
9222   }%
9223 }%
9224 }%
9225 {%
9226   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
9227 }
```

short-em-long Now the short (long) version

```
9228 \newabbreviationstyle{short-em-long}%
9229 {%
9230   \renewcommand*{\CustomAbbreviationFields}{%
9231     name={\glxtrshortlongname},
9232     sort={\the\glsshorttok},
9233     description={\the\glslongtok},%
9234     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9235       \protect\glxtrfullsep{\the\glslabeltok}%
9236       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9237     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9238       \protect\glxtrfullsep{\the\glslabeltok}%
9239       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9240     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9241   }%
```

Unset the regular attribute if it has been set.

```
9241   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9242     \glshasattribute{\the\glslabeltok}{regular}%
9243     {%
9244       \glissetattribute{\the\glslabeltok}{regular}{false}%
9245     }%
```

```

9246 {}%
9247 }%
9248 }%
9249 {%

```

Mostly as short-long style:

```

9250 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9251 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9252 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9253 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9254 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9255 \renewcommand*{\glsxtrfullformat}[2]{%
9256   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
9257   \ifglsxtrinsertinside\else##2\fi
9258   \glsxtrfullsep{##1}%
9259   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9260 }%
9261 \renewcommand*{\glsxtrfullplformat}[2]{%
9262   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
9263   \ifglsxtrinsertinside\else##2\fi
9264   \glsxtrfullsep{##1}%
9265   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9266 }%
9267 \renewcommand*{\Glsxtrfullformat}[2]{%
9268   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
9269   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9270   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9271 }%
9272 \renewcommand*{\Glsxtrfullplformat}[2]{%
9273   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
9274   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9275   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9276 }%
9277 }

```

rt-em-long-desc As before but user provides description

```

9278 \newabbreviationstyle{short-em-long-desc}%
9279 {%
9280   \renewcommand*{\CustomAbbreviationFields}{%
9281     name={\glsxtrshortlongdescname},
9282     sort={\glsxtrshortlongdescsort},
9283     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9284     \protect\glsxtrfullsep{\the\glslabeltok}}%
9285     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9286     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9287     \protect\glsxtrfullsep{\the\glslabeltok}}%
9288     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9289     text={\protect\glsabbrvemfont{\the\glsshorttok}},%

```

```

9290 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
9291 }%

```

Unset the regular attribute if it has been set.

```

9292 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9293   \glshasattribute{\the\glslabeltok}{regular}%
9294   {%
9295     \glissetattribute{\the\glslabeltok}{regular}{false}%
9296   }%
9297   {}%
9298 }%
9299 }%
9300 {%
9301   \GlsXtrUseAbbrStyleFmts{short-em-long}%
9302 }

```

hort-em-long-em

```

9303 \newabbreviationstyle{short-em-long-em}%
9304 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

9305 \renewcommand*{\CustomAbbreviationFields}{%
9306   name={\glsxtrshortlongname},
9307   sort={\the\glsshorttok},
9308   description={\protect\glslongemfont{\the\glslongtok}},%
9309   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
9310   \protect\glsxtrfullsep{\the\glslabeltok}%
9311   \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9312   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
9313   \protect\glsxtrfullsep{\the\glslabeltok}%
9314   \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9315   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9316 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9317   \glshasattribute{\the\glslabeltok}{regular}%
9318   {%
9319     \glissetattribute{\the\glslabeltok}{regular}{false}%
9320   }%
9321   {}%
9322 }%
9323 }%
9324 {%
9325   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9326   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9327   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9328   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9329   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline form are the same for this style.

```

9330 \renewcommand*{\glxtrfullformat}[2]{%
9331   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9332   \ifglxtrinsertinside\else##2\fi
9333   \glxtrfullsep{##1}%
9334   \glxtrparen{\glsfirstlongemfont{\glssaccesslong{##1}}}%
9335 }%
9336 \renewcommand*{\glxtrfullplformat}[2]{%
9337   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9338   \ifglxtrinsertinside\else##2\fi
9339   \glxtrfullsep{##1}%
9340   \glxtrparen{\glsfirstlongemfont{\glssaccesslongpl{##1}}}%
9341 }%
9342 \renewcommand*{\Glsxtrfullformat}[2]{%
9343   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9344   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9345   \glxtrparen{\glsfirstlongemfont{\glssaccesslong{##1}}}%
9346 }%
9347 \renewcommand*{\Glsxtrfullplformat}[2]{%
9348   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9349   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9350   \glxtrparen{\glsfirstlongemfont{\glssaccesslongpl{##1}}}%
9351 }%
9352 }

```

em-long-em-desc

```

9353 \newabbreviationstyle{short-em-long-em-desc}%
9354 {%
9355   \renewcommand*{\CustomAbbreviationFields}{%
9356     name={\glxtrshortlongdescname},%
9357     sort={\glxtrshortlongdescsort},%
9358     first={\protect\glsfirstabbrvemfont{\the\glssshorttok}}%
9359     \protect\glxtrfullsep{\the\glslabeltok}}%
9360     \glxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
9361     firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}}%
9362     \protect\glxtrfullsep{\the\glslabeltok}}%
9363     \glxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
9364     text={\protect\glsabbrvemfont{\the\glssshorttok}},%
9365     plural={\protect\glsabbrvemfont{\the\glssshortpltok}}}%
9366 }%

```

Unset the regular attribute if it has been set.

```

9367 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9368   \glshasattribute{\the\glslabeltok}{regular}%
9369   {%
9370     \glissetattribute{\the\glslabeltok}{regular}{false}%
9371   }%
9372   {}%
9373 }%

```

```

9374 }%
9375 {%
9376   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
9377 }

```

short-em

```

9378 \newabbreviationstyle{short-em}%
9379 {%
9380   \renewcommand*{\CustomAbbreviationFields}{%
9381     name={\glstrshortnolongname},
9382     sort={\the\glsshorttok},
9383     first={\protect\glfirstabbrvemfont{\the\glsshorttok}},
9384     firstplural={\protect\glfirstabbrvemfont{\the\glsshortpltok}},
9385     text={\protect\glabbrvemfont{\the\glsshorttok}},
9386     plural={\protect\glabbrvemfont{\the\glsshortpltok}},
9387     description={\the\glslongtok}}%
9388   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9389     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9390 }%
9391 {%
9392   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9393   \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
9394   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
9395   \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
9396   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9397   \renewcommand*{\glxtrinlinefullformat}[2]{%
9398     \protect\glfirstabbrvemfont{\glsaccessshort{##1}}%
9399     \ifglxtrininsertinside##2\fi}%
9400   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9401   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9402 }%
9403   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9404     \protect\glfirstabbrvemfont{\glsaccessshortpl{##1}}%
9405     \ifglxtrininsertinside##2\fi}%
9406   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9407   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9408 }%

9409   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9410     \protect\glfirstabbrvemfont{\Glsaccessshort{##1}}%
9411     \ifglxtrininsertinside##2\fi}%
9412   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9413   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9414 }%
9415   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9416     \protect\glfirstabbrvemfont{\Glsaccessshortpl{##1}}%
9417     \ifglxtrininsertinside##2\fi}%
9418   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9419 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9420 }%

```

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.

```

9421 \renewcommand*{\glxtrfullformat}[2]{%
9422   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9423   \ifglxtrininsertinside\else##2\fi
9424 }%
9425 \renewcommand*{\glxtrfullplformat}[2]{%
9426   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9427   \ifglxtrininsertinside\else##2\fi
9428 }%
9429 \renewcommand*{\Glsxtrfullformat}[2]{%
9430   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9431   \ifglxtrininsertinside\else##2\fi
9432 }%
9433 \renewcommand*{\Glsxtrfullplformat}[2]{%
9434   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9435   \ifglxtrininsertinside\else##2\fi
9436 }%
9437 }

```

short-em-nolong

```

9438 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

9439 \newabbreviationstyle{short-em-desc}%
9440 {%
9441   \renewcommand*{\CustomAbbreviationFields}{%
9442     name={\glxtrshortdescname},
9443     sort={\the\glssshorttok},
9444     first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},
9445     firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},
9446     text={\protect\glsabbrvemfont{\the\glssshorttok}},
9447     plural={\protect\glsabbrvemfont{\the\glssshortpltok}},
9448     description={\the\glslongtok}}%
9449   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9450     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9451 }%
9452 {%
9453   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9454   \renewcommand*{\glsabbrvfnt}[1]{\glsabbrvemfont{##1}}%
9455   \renewcommand*{\glsfirstabbrvfnt}[1]{\glsfirstabbrvemfont{##1}}%
9456   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9457   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9458 \renewcommand*{\glxtrinlinefullformat}[2]{%
9459   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%

```

```

9460     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9461     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9462 }%
9463 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9464     \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9465     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9466     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9467 }%
9468 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9469     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9470     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9471     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9472 }%
9473 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9474     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9475     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9476     \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9477 }%

```

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.

```

9478 \renewcommand*{\glxtrfullformat}[2]{%
9479     \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9480     \ifglxtrinsertinside\else##2\fi
9481 }%
9482 \renewcommand*{\glxtrfullplformat}[2]{%
9483     \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9484     \ifglxtrinsertinside\else##2\fi
9485 }%
9486 \renewcommand*{\Glsxtrfullformat}[2]{%
9487     \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9488     \ifglxtrinsertinside\else##2\fi
9489 }%
9490 \renewcommand*{\Glsxtrfullplformat}[2]{%
9491     \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9492     \ifglxtrinsertinside\else##2\fi
9493 }%
9494 }

```

-em-nolong-desc

```

9495 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9496 \newabbreviationstyle{nolong-short-em}%
9497 {%
9498     \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9499 }%
9500 {%
9501     \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```



The inline full form displays the long form followed by the short form in parentheses.

```

9502 \renewcommand*{\glxtrinlinefullformat}[2]{%
9503   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9504     \ifglxtrinsertinside##2\fi}%
9505   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9506   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9507 }%
9508 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9509   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9510     \ifglxtrinsertinside##2\fi}%
9511   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9512   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9513 }%
9514 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9515   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9516     \ifglxtrinsertinside##2\fi}%
9517   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9518   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9519 }%
9520 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9521   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9522     \ifglxtrinsertinside##2\fi}%
9523   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9524   \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9525 }%
9526 }

```

`long-noshort-em` The short form is explicitly invoked through commands like `\glsshort`.

```

9527 \newabbreviationstyle{long-noshort-em}%
9528 {%
9529   \renewcommand*{\CustomAbbreviationFields}{%
9530     name={\glxtrlongnoshortname},
9531     sort={\the\glsshorttok},
9532     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9533     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9534     text={\protect\glslongdefaultfont{\the\glslongtok}},
9535     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9536     description={\the\glslongtok}%
9537   }%
9538   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9539     \glssetattribute{\the\glslabeltok}{regular}{true}}%
9540 }%
9541 {%
9542   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9543   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9544   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9545   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9546   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9547 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9548   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9549   \ifglxtrinsertinside \else##2\fi
9550 }%
9551 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9552   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9553   \ifglxtrinsertinside \else##2\fi
9554 }%
9555 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9556   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9557   \ifglxtrinsertinside \else##2\fi
9558 }%
9559 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9560   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9561   \ifglxtrinsertinside \else##2\fi
9562 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9563 \renewcommand*{\glxtrinlinefullformat}[2]{%
9564   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9565   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9566   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9567 }%
9568 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9569   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9570   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9571   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9572 }%
9573 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9574   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9575   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9576   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9577 }%
9578 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9579   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9580   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9581   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9582 }%

```

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.

```

9583 \renewcommand*{\glxtrfullformat}[2]{%
9584   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9585   \ifglxtrinsertinside\else##2\fi
9586 }%
9587 \renewcommand*{\glxtrfullplformat}[2]{%
9588   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9589   \ifglxtrinsertinside\else##2\fi
9590 }%
9591 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9592 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9593 \ifglxtrinsertinside\else##2\fi
9594 }%
9595 \renewcommand*{\Glsxtrfullplformat}[2]{%
9596 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9597 \ifglxtrinsertinside\else##2\fi
9598 }%
9599 }

```

long-em Backward compatibility:

```

9600 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9601 \newabbreviationstyle{long-em-noshort-em}%
9602 {%
9603 \renewcommand*{\CustomAbbreviationFields}{%
9604 name={\glxtrlongnoshortname},
9605 sort={\the\glsshorttok},
9606 first={\protect\glsfirstlongemfont{\the\glslongtok}},
9607 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9608 text={\protect\glslongemfont{\the\glslongtok}},
9609 plural={\protect\glslongemfont{\the\glslongpltok}},%
9610 description={\protect\glslongemfont{\the\glslongtok}}%
9611 }%
9612 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9613 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9614 }%
9615 {%
9616 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9617 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9618 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9619 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9620 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9621 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9622 \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9623 \ifglxtrinsertinside \else##2\fi
9624 }%
9625 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9626 \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9627 \ifglxtrinsertinside \else##2\fi
9628 }%
9629 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9630 \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9631 \ifglxtrinsertinside \else##2\fi
9632 }%
9633 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9634 \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9635 \ifglxtrinsertinside \else##2\fi

```

9636 }%

The inline full form displays the long format followed by the short form in parentheses.

```
9637 \renewcommand*{\glxstrinlinefullformat}[2]{%
9638   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9639   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9640   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9641 }%
9642 \renewcommand*{\glxstrinlinefullplformat}[2]{%
9643   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9644   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9645   \glxstrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9646 }%
9647 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9648   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9649   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9650   \glxstrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9651 }%
9652 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9653   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9654   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9655   \glxstrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9656 }%
```

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.

```
9657 \renewcommand*{\glxstrfullformat}[2]{%
9658   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9659   \ifglxstrinsertinside\else##2\fi
9660 }%
9661 \renewcommand*{\glxstrfullplformat}[2]{%
9662   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9663   \ifglxstrinsertinside\else##2\fi
9664 }%
9665 \renewcommand*{\Glsxtrfullformat}[2]{%
9666   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxstrinsertinside##2\fi}%
9667   \ifglxstrinsertinside\else##2\fi
9668 }%
9669 \renewcommand*{\Glsxtrfullplformat}[2]{%
9670   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxstrinsertinside##2\fi}%
9671   \ifglxstrinsertinside\else##2\fi
9672 }%
9673 }
```

oshort-em-noreg Like long-em-noshort-em but doesn't set the regular attribute.

```
9674 \newabbreviationstyle{long-em-noshort-em-noreg}%
9675 {%
9676   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
```

Unset the regular attribute if it has been set.

```

9677 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9678   \glshasattribute{\the\glslabeltok}{regular}}%
9679   {%
9680     \glissetattribute{\the\glslabeltok}{regular}{false}}%
9681   }%
9682   {}%
9683 }%
9684 }%
9685 {%
9686   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}}%
9687 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9688 \newabbreviationstyle{long-noshort-em-desc}%
9689 {%
9690   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}}%
9691 }%
9692 {%
9693   \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
9694   \renewcommand*{\glssabrvfont}[1]{\glssabrvemfont{##1}}%
9695   \renewcommand*{\glsfirstabrvfont}[1]{\glsfirstabrvemfont{##1}}%
9696   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9697   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9698 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9699   \glslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9700   \ifglsxtrininsertinside \else##2\fi
9701 }%
9702 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9703   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9704   \ifglsxtrininsertinside \else##2\fi
9705 }%
9706 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9707   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9708   \ifglsxtrininsertinside \else##2\fi
9709 }%
9710 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9711   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9712   \ifglsxtrininsertinside \else##2\fi
9713 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9714 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9715   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9716   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9717   \glsxtrparen{\protect\glsfirstabrvemfont{\glssaccessshort{##1}}}%
9718 }%
9719 \renewcommand*{\glsxtrinlinefullplformat}[2]{%

```

```

9720 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9721 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9722 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9723 }%
9724 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9725 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9726 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9727 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9728 }%
9729 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9730 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9731 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9732 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9733 }%

```

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.

```

9734 \renewcommand*{\glxtrfullformat}[2]{%
9735 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9736 \ifglxtrinsertinside\else##2\fi
9737 }%
9738 \renewcommand*{\glxtrfullplformat}[2]{%
9739 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9740 \ifglxtrinsertinside\else##2\fi
9741 }%
9742 \renewcommand*{\Glsxtrfullformat}[2]{%
9743 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9744 \ifglxtrinsertinside\else##2\fi
9745 }%
9746 \renewcommand*{\Glsxtrfullplformat}[2]{%
9747 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9748 \ifglxtrinsertinside\else##2\fi
9749 }%
9750 }

```

long-desc-em Backward compatibility:

```
9751 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

noshort-em-desc The short form is explicitly invoked through commands like `\glsshort`. The long form is emphasized.

```

9752 \newabbreviationstyle{long-em-noshort-em-desc}%
9753 {%
9754 \renewcommand*{\CustomAbbreviationFields}{%
9755 name={\glxtrlongnoshortdescname},
9756 sort={\the\glslongtok},
9757 first={\protect\glsfirstlongemfont{\the\glslongtok}},
9758 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9759 text={\glslongemfont{\the\glslongtok}},
9760 plural={\glslongemfont{\the\glslongpltok}}}%

```

```

9761 }%
9762 \renewcommand*\GlsXtrPostNewAbbreviation}{%
9763   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9764 }%
9765 {%
9766 \renewcommand*\abbrvpluralsuffix{\protect\glsxtremsuffix}%
9767 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9768 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9769 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
9770 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9771 \renewcommand*\glsxtrsubsequentfmt}[2]{%
9772   \glslongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9773   \ifglsxtrininsertinside \else##2\fi
9774 }%
9775 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
9776   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9777   \ifglsxtrininsertinside \else##2\fi
9778 }%
9779 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
9780   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9781   \ifglsxtrininsertinside \else##2\fi
9782 }%
9783 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
9784   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9785   \ifglsxtrininsertinside \else##2\fi
9786 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9787 \renewcommand*\glsxtrinlinefullformat}[2]{%
9788   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9789   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9790   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9791 }%
9792 \renewcommand*\glsxtrinlinefullplformat}[2]{%
9793   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9794   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9795   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9796 }%
9797 \renewcommand*\Glsxtrinlinefullformat}[2]{%
9798   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9799   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9800   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9801 }%
9802 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9803   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9804   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9805   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9806 }%

```

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.

```

9807 \renewcommand*{\glxtrfullformat}[2]{%
9808   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9809   \ifglxtrinsertinside\else##2\fi
9810 }%
9811 \renewcommand*{\glxtrfullplformat}[2]{%
9812   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9813   \ifglxtrinsertinside\else##2\fi
9814 }%
9815 \renewcommand*{\Glsxtrfullformat}[2]{%
9816   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9817   \ifglxtrinsertinside\else##2\fi
9818 }%
9819 \renewcommand*{\Glsxtrfullplformat}[2]{%
9820   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9821   \ifglxtrinsertinside\else##2\fi
9822 }%
9823 }

```

`t-em-desc-noreg` Like `long-em-noshort-em-desc` but doesn't set the regular attribute.

```

9824 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
9825 {%
9826   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%

```

Unset the regular attribute if it has been set.

```

9827 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9828   \glshasattribute{\the\glslabeltok}{regular}%
9829   {%
9830     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
9831   }%
9832   {}%
9833 }%
9834 }%
9835 {%
9836   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
9837 }

```

`ort-em-footnote`

```

9838 \newabbreviationstyle{short-em-footnote}%
9839 {%
9840   \renewcommand*{\CustomAbbreviationFields}{%
9841     name={\glxtrfootnotename},
9842     sort={\the\glsshorttok},
9843     description={\the\glslongtok},%
9844     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9845       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9846       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9847     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%

```



```

9848 \protect\glstrabbrvfootnote{\the\glslabeltok}%
9849 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9850 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

9851 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9852 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9853 \glsattribute{\the\glslabeltok}{regular}%
9854 {%
9855 \glssetattribute{\the\glslabeltok}{regular}{false}%
9856 }%
9857 }%
9858 }%
9859 }%
9860 {%
9861 \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
9862 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9863 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9864 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9865 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9866 \renewcommand*{\glstrfullformat}[2]{%
9867 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9868 \ifglstrinsertinside\else##2\fi
9869 \protect\glstrabbrvfootnote{##1}%
9870 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9871 }%
9872 \renewcommand*{\glstrfullplformat}[2]{%
9873 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9874 \ifglstrinsertinside\else##2\fi
9875 \protect\glstrabbrvfootnote{##1}%
9876 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9877 }%
9878 \renewcommand*{\GlsXtrfullformat}[2]{%
9879 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9880 \ifglstrinsertinside\else##2\fi
9881 \protect\glstrabbrvfootnote{##1}%
9882 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9883 }%
9884 \renewcommand*{\GlsXtrfullplformat}[2]{%
9885 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9886 \ifglstrinsertinside\else##2\fi
9887 \protect\glstrabbrvfootnote{##1}%
9888 {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9889 }%

```

The first use full form and the inline full form use the short (long) style.

```

9890 \renewcommand*{\glstrinlinefullformat}[2]{%
9891 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%

```

```

9892 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9893 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9894 }%
9895 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9896 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9897 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9898 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9899 }%
9900 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9901 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9902 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9903 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9904 }%
9905 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9906 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9907 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9908 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9909 }%
9910 }

```

footnote-em Backward compatibility:

```

9911 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

9912 \newabbreviationstyle{short-em-postfootnote}%
9913 {%
9914 \renewcommand*{\CustomAbbreviationFields}{%
9915 name={\glxtrfootnotename},
9916 sort={\the\glssshorttok},
9917 description={\the\glslongtok},%
9918 first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},%
9919 firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},%
9920 plural={\protect\glssabbrvemfont{\the\glssshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

9921 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9922 \csdef{glxtrpostlink\glscategorylabel}{%
9923 \glxtrifwasfirstuse
9924 }%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9925 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9926 {\glsfirstlongfootnotefont{\glssentrylong{\glslabel}}}%
9927 }%
9928 {}%
9929 }%
9930 \glshasattribute{\the\glslabeltok}{regular}%
9931 {%

```

```

9932     \glsetattribute{\the\glslabeltok}{regular}{false}%
9933   }%
9934   {}%
9935 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9936 \renewcommand*\glxtrsetupfulldefs{%
9937   \let\glxtrifwasfirstuse\@secondoftwo
9938 }%
9939}%
9940{%
9941 \renewcommand*\abbrvpluralsuffix{\protect\glxtremsuffix}%
9942 \renewcommand*\glabbrvfont[1]{\glabbrvemfont{##1}}%
9943 \renewcommand*\glfirstabbrvfont[1]{\glfirstabbrvemfont{##1}}%
9944 \renewcommand*\glfirstlongfont[1]{\glfirstlongfootnotefont{##1}}%
9945 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9946 \renewcommand*\glxtrfullformat[2]{%
9947   \glfirstabbrvemfont{\glaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9948   \ifglxtrininsertinside\else##2\fi
9949 }%
9950 \renewcommand*\glxtrfullplformat[2]{%
9951   \glfirstabbrvemfont{\glaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9952   \ifglxtrininsertinside\else##2\fi
9953 }%
9954 \renewcommand*\Glsxtrfullformat[2]{%
9955   \glfirstabbrvemfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9956   \ifglxtrininsertinside\else##2\fi
9957 }%
9958 \renewcommand*\Glsxtrfullplformat[2]{%
9959   \glfirstabbrvemfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9960   \ifglxtrininsertinside\else##2\fi
9961 }%

```

The first use full form and the inline full form use the short (long) style.

```

9962 \renewcommand*\glxtrininlinefullformat[2]{%
9963   \glfirstabbrvemfont{\glaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9964   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9965   \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%
9966 }%
9967 \renewcommand*\glxtrininlinefullplformat[2]{%
9968   \glfirstabbrvemfont{\glaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9969   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9970   \glxtrparen{\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
9971 }%
9972 \renewcommand*\Glsxtrininlinefullformat[2]{%
9973   \glfirstabbrvemfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9974   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9975   \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%

```

```

9976 }%
9977 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9978   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9979   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9980   \glsxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9981 }%
9982 }

```

postfootnote-em Backward compatibility:

```

9983 \@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.

```

9984 \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.

```

9985 \ifdef\glscurrentfieldvalue
9986 {
9987   \newcommand*{\glsxtruserparen}[2]{%
9988     \glsxtrfullsep{#2}%
9989     \glsxtrparen
9990     {#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%
9991   }
9992 }
9993 {
9994   \newcommand*{\glsxtruserparen}[2]{%
9995     \glsxtrfullsep{#2}%
9996     \glsxtrparen
9997     {#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{}}%
9998   }
9999 }

```

Font used for short form:

lsabbrvuserfont

```

10000 \newcommand*{\glsabbrvuserfont}[1]{\glsabbrvdefaultfont{#1}}

```

Font used for short form on first use:

stabbrvuserfont

```

10001 \newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}

```

Font used for long form:

glslonguserfont

```
10002 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
10003 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
10004 \newcommand*{\lsxtrusersuffix}{\lsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
10005 \newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
10006 \newabbreviationstyle{long-short-user}%
10007 {%
10008   \renewcommand*{\CustomAbbreviationFields}{%
10009     name={\glxtrlongshortname},
10010     sort={\the\glsshorttok},
10011     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10012       \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
10013       {\the\glslabeltok}},%
10014     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10015       \protect\glxtruserparen
10016       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10017     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10018     description={\protect\glsuserdescription{\the\glslongtok}%
10019       {\the\glslabeltok}}}%
10019 }
```

Unset the regular attribute if it has been set.

```
10020 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10021   \glshasattribute{\the\glslabeltok}{regular}%
10022   {%
10023     \glissetattribute{\the\glslabeltok}{regular}{false}%
10024   }%
10025   {}%
10026 }%
10027 }%
10028 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
10029 \renewcommand*{\abbrvpluralsuffix}{\lsxtrusersuffix}%
10030 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10031 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10032 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10033 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline form are the same for this style.

```

10034 \renewcommand*{\glxtrfullformat}[2]{%
10035   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10036   \ifglxtrinsertinside\else##2\fi
10037   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10038 }%
10039 \renewcommand*{\glxtrfullplformat}[2]{%
10040   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10041   \ifglxtrinsertinside\else##2\fi
10042   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10043 }%
10044 \renewcommand*{\Glsxtrfullformat}[2]{%
10045   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10046   \ifglxtrinsertinside\else##2\fi
10047   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
10048 }%
10049 \renewcommand*{\Glsxtrfullplformat}[2]{%
10050   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10051   \ifglxtrinsertinside\else##2\fi
10052   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
10053 }%
10054 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

10055 \newabbreviationstyle{long-postshort-user}%
10056 {%
10057   \renewcommand*{\CustomAbbreviationFields}{%
10058     name={\glxtrlongshortname},
10059     sort={\the\glsshorttok},
10060     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10061     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
10062     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10063     description={\protect\glsuserdescription{\the\glslongtok}%
10064       {\the\glslabeltok}}}%
10065   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10066     \csdef{glxtrpostlink\glscategorylabel}{%
10067       \glxtrifwasfirstuse
10068       {%
10069         \glxtruserparen
10070         {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10071         {\glslabel}%
10072       }%
10073     }%
10074   }%
10075   \glsasattribute{\the\glslabeltok}{regular}%
10076   {%
10077     \glssetattribute{\the\glslabeltok}{regular}{false}%
10078   }%

```

```

10079    {}%
10080  }%
10081 }%
10082 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10083 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10084 \renewcommand*{\glsabrvfont}[1]{\glsabrvuserfont{##1}}%
10085 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10086 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10087 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10088 \renewcommand*{\glxtrfullformat}[2]{%
10089   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10090   \ifglxtrininsertinside\else##2\fi
10091 }%
10092 \renewcommand*{\glxtrfullplformat}[2]{%
10093   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10094   \ifglxtrininsertinside\else##2\fi
10095 }%
10096 \renewcommand*{\Glsxtrfullformat}[2]{%
10097   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10098   \ifglxtrininsertinside\else##2\fi
10099 }%
10100 \renewcommand*{\Glsxtrfullplformat}[2]{%
10101   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10102   \ifglxtrininsertinside\else##2\fi
10103 }%

```

In-line format:

```

10104 \renewcommand*{\glxtrinlinefullformat}[2]{%
10105   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10106   \ifglxtrininsertinside\else##2\fi
10107   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10108 }%
10109 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10110   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10111   \ifglxtrininsertinside\else##2\fi
10112   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
10113 }%
10114 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10115   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10116   \ifglxtrininsertinside\else##2\fi
10117   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
10118 }%
10119 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10120   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10121   \ifglxtrininsertinside\else##2\fi
10122   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%

```

```

10123 }%
10124 }

```

ortuserdescname

```

10125 \newcommand*{\glxtrlongshortuserdescname}{%
10126   \protect\glslonguserfont{\the\glslongtok}%
10127   \protect\glxtruserparen
10128   {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
10129 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

10130 \newabbreviationstyle{long-postshort-user-desc}%
10131 {%
10132   \renewcommand*{\CustomAbbreviationFields}{%
10133     name={\glxtrlongshortuserdescname},
10134     sort={\the\glslongtok},
10135     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10136     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10137     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
10138     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
10139 }%
10140 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10141   \csdef{glxtrpostlink\glscategorylabel}{%
10142     \glxtrifwasfirstuse
10143     {%
10144       \glxtruserparen
10145       {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
10146       {\glslabel}%
10147     }%
10148     {}%
10149   }%
10150   \glshasattribute{\the\glslabeltok}{regular}%
10151   {%
10152     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10153   }%
10154   {}%
10155 }%
10156 }%
10157 {%
10158   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
10159 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

10160 \newabbreviationstyle{short-postlong-user}%
10161 {%
10162   \renewcommand*{\CustomAbbreviationFields}{%
10163     name={\glxtrshortlongname},
10164     sort={\the\glsshorttok},

```



```

10165     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10166     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10167     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
10168     description={\protect\glsuserdescription{\the\glslongtok}%
10169       {\the\glslabeltok}}}%
10170 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10171   \csdef{glsxtrpostlink\glscategorylabel}{%
10172     \glsxtrifwasfirstuse
10173     {%
10174       \glsxtruserparen
10175       {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10176       {\glslabel}%
10177     }%
10178   }%
10179 }%
10180 \glsattribute{\the\glslabeltok}{regular}%
10181 {}%
10182 \glssetattribute{\the\glslabeltok}{regular}{false}%
10183 }%
10184 {}%
10185 }%
10186 }%
10187 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10188 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
10189 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
10190 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
10191 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
10192 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

10193 \renewcommand*{\glsxtrfullformat}[2]{%
10194   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10195   \ifglsxtrininsertinside\else##2\fi
10196 }%
10197 \renewcommand*{\glsxtrfullplformat}[2]{%
10198   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10199   \ifglsxtrininsertinside\else##2\fi
10200 }%
10201 \renewcommand*{\Glsxtrfullformat}[2]{%
10202   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10203   \ifglsxtrininsertinside\else##2\fi
10204 }%
10205 \renewcommand*{\Glsxtrfullplformat}[2]{%
10206   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10207   \ifglsxtrininsertinside\else##2\fi
10208 }%

```

In-line format:

```

10209 \renewcommand*{\glxtrinlinefullformat}[2]{%
10210   \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10211   \ifglxtrininsertinside\else##2\fi
10212   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
10213 }%
10214 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10215   \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10216   \ifglxtrininsertinside\else##2\fi
10217   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10218 }%
10219 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10220   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10221   \ifglxtrininsertinside\else##2\fi
10222   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10223 }%
10224 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10225   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10226   \ifglxtrininsertinside\else##2\fi
10227   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
10228 }%
10229 }

```

onguserdesname

```

10230 \newcommand*{\glxtrshortlonguserdesname}{%
10231   \protect\glssabbrvuserfont{\the\glssshorttok}%
10232   \protect\glxtruserparen
10233     {\protect\glslonguserfont{\the\glslongpltok}}}%
10234     {\the\glslabelltok}%
10235 }

```

tlong-user-desc Like short-postlong-user but leaves the user to specify the description.

```

10236 \newabbreviationstyle{short-postlong-user-desc}%
10237 {%
10238   \renewcommand*{\CustomAbbreviationFields}{%
10239     name={\glxtrshortlonguserdesname},
10240     sort={\the\glssshorttok},
10241     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
10242     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

10243     text={\protect\glssabbrvuserfont{\the\glssshorttok}},%
10244     plural={\protect\glssabbrvuserfont{\the\glssshortpltok}}}%
10245 }%
10246 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10247   \csdef{glxtrpostlink\glscategorylabel}{%
10248     \glxtrifwasfirstuse
10249     {%
10250       \glxtruserparen
10251         {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
10252         {\glslabel}%

```

```

10253     }%
10254     {}%
10255     }%
10256     \glshasattribute{\the\glslabeltok}{regular}%
10257     {%
10258         \glissetattribute{\the\glslabeltok}{regular}{false}%
10259     }%
10260     {}%
10261 }%
10262 }%
10263 {%
10264     \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
10265 }

```

short-user-desc

```

10266 \newabbreviationstyle{long-short-user-desc}%
10267 {%
10268     \renewcommand*{\CustomAbbreviationFields}{%
10269         name={\glsxtrlongshortuserdescname},
10270         sort={\glsxtrlongshortdescsort},%

10271         first={\protect\glsfirstlonguserfont{\the\glslongtok}%
10272             \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
10273             {\the\glslabeltok}},%
10274         firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
10275             \protect\glsxtruserparen
10276             {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
10277         text={\protect\glsabbrvfont{\the\glsshorttok}},%
10278         plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
10279     }%

```

Unset the regular attribute if it has been set.

```

10280 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10281     \glshasattribute{\the\glslabeltok}{regular}%
10282     {%
10283         \glissetattribute{\the\glslabeltok}{regular}{false}%
10284     }%
10285     {}%
10286 }%
10287 }%
10288 {%
10289     \GlsXtrUseAbbrStyleFmts{long-short-user}%
10290 }

```

short-long-user

```

10291 \newabbreviationstyle{short-long-user}%
10292 {%

```

\glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in \glsuserdescription.)

```

10293 \renewcommand*{\CustomAbbreviationFields}{%
10294     name={\glxtrshortlongname},
10295     sort={\the\glsshorttok},
10296     description={\protect\gluserdescription{\the\glslongtok}%
10297         {\the\glslabeltok}},%
10298     first={\protect\glfirstabbrvuserfont{\the\glsshorttok}%
10299         \protect\glxtruserparen{\protect\glfirstlonguserfont{\the\glslongtok}}%
10300         {\the\glslabeltok}},%
10301     firstplural={\protect\glfirstabbrvuserfont{\the\glsshortpltok}%
10302         \protect\glxtruserparen{\protect\glfirstlonguserfont{\the\glslongpltok}}%
10303         {\the\glslabeltok}},%
10304     plural={\protect\glabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10305 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10306     \glshasattribute{\the\glslabeltok}{regular}%
10307     {%
10308         \glissetattribute{\the\glslabeltok}{regular}{false}%
10309     }%
10310     {}%
10311 }%
10312 }%
10313 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10314 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
10315 \renewcommand*{\glabbrvfont}[1]{\glabbrvuserfont{##1}}%
10316 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvuserfont{##1}}%
10317 \renewcommand*{\glfirstlongfont}[1]{\glfirstlonguserfont{##1}}%
10318 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10319 \renewcommand*{\glxtrfullformat}[2]{%
10320     \glfirstabbrvuserfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10321     \ifglxtrininsertinside\else##2\fi
10322     \glxtruserparen{\glfirstlonguserfont{\glaccesslong{##1}}}{##1}%
10323 }%
10324 \renewcommand*{\glxtrfullplformat}[2]{%
10325     \glfirstabbrvuserfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10326     \ifglxtrininsertinside\else##2\fi
10327     \glxtruserparen{\glfirstlonguserfont{\glaccesslongpl{##1}}}{##1}%
10328 }%
10329 \renewcommand*{\Glsxtrfullformat}[2]{%
10330     \glfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10331     \ifglxtrininsertinside\else##2\fi
10332     \glxtruserparen{\glfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
10333 }%
10334 \renewcommand*{\Glsxtrfullplformat}[2]{%
10335     \glfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10336     \ifglxtrininsertinside\else##2\fi

```

```

10337   \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
10338 }%
10339 }

```

-long-user-desc

```

10340 \newabbreviationstyle{short-long-user-desc}%
10341 {%
10342   \renewcommand*{\CustomAbbreviationFields}{%
10343     name={\glxtrshortlonguserdescname},
10344     sort={\glxtrshortlongdescsort},%

10345     first={\protect\glsfirstabbrvuserfont{\the\glssshorttok}}%
10346     \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
10347     {\the\glslabeltok}},%
10348     firstplural={\protect\glsfirstabbrvuserfont{\the\glssshortpltok}}%
10349     \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
10350     {\the\glslabeltok}},%
10351     text={\protect\glsabbrvfont{\the\glssshorttok}},%
10352     plural={\protect\glsabbrvfont{\the\glssshortpltok}}}%
10353 }%

```

Unset the regular attribute if it has been set.

```

10354 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10355   \glshasattribute{\the\glslabeltok}{regular}%
10356   {%
10357     \glissetattribute{\the\glslabeltok}{regular}{false}%
10358   }%
10359   {}%
10360 }%
10361 }%
10362 {%
10363   \GlsXtrUseAbbrStyleFmts{short-long-user}%
10364 }

```

### 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.

```

10365 \newrobustcmd*{\glxtrifhyphenstart}[3]{%
10366   \ifx\glsinsert#1\relax
10367     \expandafter@\glxtrifhyphenstart#1\relax\relax
10368     \@end@glxtrifhyphenstart{#2}{#3}%
10369   \else
10370     \@glxtrifhyphenstart#1\relax\relax\@end@glxtrifhyphenstart{#2}{#3}%

```

```
10371 \fi
10372 }
```

trifhyphenstart

```
10373 \def\@glxstrifhyphenstart#1#2\end@glxstrifhyphenstart#3#4{%
10374 \ifx-#1\relax#3\else #4\fi
10375 }
```

rlonghyphenshort

```
\glxstrlonghyphenshort{<label>}{<long>}{<short>}{<insert>}
```

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be the first letter uppercase form.

```
10376 \newcommand*\glxstrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10377 {%
```

If *<insert>* starts with a hyphen, redefine `\glxstrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glxstrwordsep` if *<insert>* doesn't start with a hyphen.

```
10378 \glxstrifhyphenstart{#4}{\def\glxstrwordsep{-}}{}%
10379 \glxfirstlonghyphenfont{#2\ifglxstrinsertinside{#4}\fi}%
10380 \ifglxstrinsertinside\else{#4}\fi
10381 \glxstrfullsep{#1}%
10382 \glxstrparen{\glxfirstabbrvhyphenfont{#3\ifglxstrinsertinside{#4}\fi}%
10383 \ifglxstrinsertinside\else{#4}\fi}%
10384 }%
10385 }
```

abbrvhyphenfont

```
10386 \newcommand*\glxabbrvhyphenfont{\glxabbrvdefaultfont}%
```

abbrvhyphenfont

```
10387 \newcommand*\glxfirstabbrvhyphenfont{\glxabbrvhyphenfont}%
```

slonghyphenfont

```
10388 \newcommand*\glxslonghyphenfont{\glxslongdefaultfont}%
```

tlonghyphenfont

```
10389 \newcommand*\glxfirstlonghyphenfont{\glxslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
10390 \newcommand*\glxstrhyphensuffix{\glxstrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the keywords attribute.

```
10391 \newabbreviationstyle{long-hyphen-short-hyphen}%
10392 {%
10393   \renewcommand*{\CustomAbbreviationFields}{%
10394     name={\glxtrlongshortname},
10395     sort={\the\glsshorttok},
10396     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10397       \protect\glxtrfullsep{\the\glslabeltok}%
10398       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10399     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10400       \protect\glxtrfullsep{\the\glslabeltok}%
10401       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10402     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}},%
10403     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10404 }
```

Unset the regular attribute if it has been set.

```
10404 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10405   \glshasattribute{\the\glslabeltok}{regular}%
10406   {%
10407     \glissetattribute{\the\glslabeltok}{regular}{false}%
10408   }%
10409 }%
10410 }%
10411 }%
10412 {%
10413   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10414   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10415   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10416   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10417   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
10418 }
```

The first use full form and the inline full form are the same for this style.

```
10418 \renewcommand*{\glxtrfullformat}[2]{%
10419   \glxtrlonghyphenshort{##1}{\glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
10420 }%
10421 \renewcommand*{\glxtrfullplformat}[2]{%
10422   \glxtrlonghyphenshort{##1}{\glsaccesslongpl{##1}}%
10423   {\glsaccessshortpl{##1}}{##2}%
10424 }%
10425 \renewcommand*{\Glsxtrfullformat}[2]{%
10426   \glxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\Glsaccessshort{##1}}{##2}%
10427 }%
10428 \renewcommand*{\Glsxtrfullplformat}[2]{%
10429   \glxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
10430   {\Glsaccessshortpl{##1}}{##2}%
10431 }%
10432 }
```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```
10433 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
10434 }
```

```

10434 {%
10435   \renewcommand*{\CustomAbbreviationFields}{%
10436     name={\glxtrlongshortdescname},
10437     sort={\glxtrlongshortdescsort},
10438     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
10439       \protect\glxtrfullsep{\the\glslabeltok}%
10440       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
10441     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
10442       \protect\glxtrfullsep{\the\glslabeltok}%
10443       \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
10444     text={\protect\glssabbvrhyphenfont{\the\glsshorttok}},%
10445     plural={\protect\glssabbvrhyphenfont{\the\glsshortpltok}}%
10446   }%

```

Unset the regular attribute if it has been set.

```

10447   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10448     \glshasattribute{\the\glslabeltok}{regular}%
10449     {%
10450       \glissetattribute{\the\glslabeltok}{regular}{false}%
10451     }%
10452   }%
10453 }%
10454 }%
10455 {%
10456   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10457 }

```

onghyphennoshort

```
\glxtrlonghyphennoshort{<label>}{<long>}{<insert>}
```

```

10458 \newcommand*{\glxtrlonghyphennoshort}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10459 {%

```

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.

```

10460   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
10461     \glsfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
10462     \ifglxtrininsertinsideelse{#3}\fi
10463   }%
10464 }

```

short-desc-noreg

This version doesn't show the short form (except explicitly with `\glxtrshort`). Since `\glxtrshort` doesn't support the hyphen switch, the short form just uses the default short-form font command. This style won't work with the regular as the regular form isn't flexible enough.



```

10465 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10466 {%
10467   \renewcommand*{\CustomAbbreviationFields}{%
10468     name={\glxtrlongnoshortdescname},
10469     sort={\expandonce\glxtrorglong},
10470     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10471     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10472     plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
10473   }%

```

Unset the regular attribute if it has been set.

```

10474   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10475     \glshasattribute{\the\glslabeltok}{regular}%
10476     {%
10477       \glissetattribute{\the\glslabeltok}{regular}{false}%
10478     }%
10479   }%
10480 }%
10481 }%
10482 {%
10483   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10484   \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10485   \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
10486   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
10487   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10488   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10489   \renewcommand*{\glxtrsubsequentfmt}[2]{%
10490     \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10491   }%
10492   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10493     \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10494   }%
10495   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10496     \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10497   }%
10498   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10499     \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10500   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10501   \renewcommand*{\glxtrininlinefullformat}[2]{%
10502     \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10503     \glxtrfullsep{##1}%
10504     \glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{##1}}}%
10505   }%
10506   \renewcommand*{\glxtrininlinefullplformat}[2]{%
10507     \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%

```

```

10508 \glstrfullsep{##1}%
10509 \glstrparen{\protect\glfirstabbrvfont{\glaccessshortpl{##1}}}%
10510 }%
10511 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10512 \glstrlonghyphennohshort{##1}{\glaccesslong{##1}}{##2}%
10513 \glstrfullsep{##1}%
10514 \glstrparen{\protect\glfirstabbrvfont{\glaccessshort{##1}}}%
10515 }%
10516 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10517 \glstrlonghyphennohshort{##1}{\glaccesslongpl{##1}}{##2}%
10518 \glstrfullsep{##1}%
10519 \glstrparen{\protect\glfirstabbrvfont{\glaccessshortpl{##1}}}%
10520 }%

```

The first use full form only displays the long form.

```

10521 \renewcommand*{\glstrfullformat}[2]{%
10522 \glstrlonghyphennohshort{##1}{\glaccesslong{##1}}{##2}%
10523 }%
10524 \renewcommand*{\glstrfullplformat}[2]{%
10525 \glstrlonghyphennohshort{##1}{\glaccesslongpl{##1}}{##2}%
10526 }%
10527 \renewcommand*{\Glsxtrfullformat}[2]{%
10528 \glstrlonghyphennohshort{##1}{\glaccesslong{##1}}{##2}%
10529 }%
10530 \renewcommand*{\Glsxtrfullplformat}[2]{%
10531 \glstrlonghyphennohshort{##1}{\glaccesslongpl{##1}}{##2}%
10532 }%
10533 }

```

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.

```

10534 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10535 {%
10536 \renewcommand*{\CustomAbbreviationFields}{%
10537 name={\glstrlongnoshortname},
10538 sort={\the\glsshorttok},
10539 first={\protect\glfirstlonghyphenfont{\the\glslongtok}},%
10540 firstplural={\protect\glfirstlonghyphenfont{\the\glslongpltok}},%
10541 text={\protect\glslonghyphenfont{\the\glslongtok}},%
10542 plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
10543 description={\the\glslongtok}%
10544 }%

```

Unset the regular attribute if it has been set.

```

10545 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10546 \glshasattribute{\the\glslabeltok}{regular}%
10547 {%
10548 \glissetattribute{\the\glslabeltok}{regular}{false}%
10549 }%

```

```

10550     {}%
10551   }%
10552 }%
10553 {%
10554   \GlsXtrUseAbbrStyleFmts{long-desc}%
10555 }

```

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.

```

10556 \newcommand*{\glsxtrlonghyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10557   {%
10558     \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
10559     \glsfirstlonghyphenfont{#1}%
10560   }%
10561 }

```

rposthyphenshort `\glsxtrposthyphenshort{<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.

```

10562 \newcommand*{\glsxtrposthyphenshort}[2]{%
10563   {%
10564     \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
10565     \ifglsxtrininsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
10566     \glsxtrfullsep{#1}%
10567     \glsxtrparen
10568     {\glsfirstabbrvhyphenfont{\glsentryshort{#1}}\ifglsxtrininsertinside{#2}\fi}%
10569     \ifglsxtrininsertinside\else{#2}\fi
10570   }%
10571 }%
10572 }

```

hyphensequent `\glsxtrposthyphensequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10573 \newcommand*{\glsxtrposthyphensequent}[2]{%
10574   \glsabbrvfont{\ifglsxtrininsertinside {#2}\fi}%

```

```

10575 \ifglxtrinsertinside \else{#2}\fi
10576 }

```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10577 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10578 {%
10579   \renewcommand*{\CustomAbbreviationFields}{%
10580     name={\glxtrlongshortname},
10581     sort={\the\glsshorttok},
10582     first={\protect\glxtrfirstlonghyphenfont{\the\glslongtok}},%
10583     firstplural={\protect\glxtrfirstlonghyphenfont{\the\glslongpltok}},%
10584     plural={\protect\glxtrabbrvhyphenfont{\the\glsshortpltok}},%
10585     description={\protect\glxtrlonghyphenfont{\the\glslongtok}}}%
10586   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10587     \csdef{glxtrpostlink\glscategorylabel}{%
10588       \glxtrifwasfirstuse
10589       {%
10590         \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10591       }%
10592     }%

```

Put the insertion into the post-link:

```

10593       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10594     }%
10595   }%
10596   \glshasattribute{\the\glslabeltok}{regular}%
10597   {%
10598     \glissetattribute{\the\glslabeltok}{regular}{false}%
10599   }%
10600   {}%
10601 }%
10602 }%
10603 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10604 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10605 \renewcommand*{\glxtrabbrvfont}[1]{\glxtrabbrvhyphenfont{##1}}%
10606 \renewcommand*{\glxtrfirstabbrvfont}[1]{\glxtrfirstabbrvhyphenfont{##1}}%
10607 \renewcommand*{\glxtrfirstlongfont}[1]{\glxtrfirstlonghyphenfont{##1}}%
10608 \renewcommand*{\glxtrlongfont}[1]{\glxtrlonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10609 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10610   \glxtrabbrvfont{\glxtraccessshort{##1}}%
10611 }%
10612 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10613   \glxtrabbrvfont{\glxtraccessshortpl{##1}}%
10614 }%
10615 \renewcommand*{\GlsXtrsubsequentfmt}[2]{%

```

```

10616 \glsabbrvfont{\Glsaccessshort{##1}}%
10617 }%
10618 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10619 \glsabbrvfont{\Glsaccessshortpl{##1}}%
10620 }%

First use full form:

10621 \renewcommand*{\glsxtrfullformat}[2]{%
10622 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10623 }%
10624 \renewcommand*{\glsxtrfullplformat}[2]{%
10625 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10626 }%
10627 \renewcommand*{\Glsxtrfullformat}[2]{%
10628 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10629 }%
10630 \renewcommand*{\Glsxtrfullplformat}[2]{%
10631 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10632 }%

```

#### In-line format.

```

10633 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10634 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10635 \ifglsxtrininsertinside{##2}\fi}%
10636 \ifglsxtrininsertinside \else{##2}\fi
10637 }%
10638 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10639 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10640 \ifglsxtrininsertinside{##2}\fi}%
10641 \ifglsxtrininsertinside \else{##2}\fi
10642 }%
10643 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10644 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
10645 \ifglsxtrininsertinside{##2}\fi}%
10646 \ifglsxtrininsertinside \else{##2}\fi
10647 }%
10648 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10649 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10650 \ifglsxtrininsertinside{##2}\fi}%
10651 \ifglsxtrininsertinside \else{##2}\fi
10652 }%
10653 }

```

`ort-hyphen-desc` Like `long-hyphen-postshort-hyphen` but the description must be supplied by the user.

```

10654 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10655 {%
10656 \renewcommand*{\CustomAbbreviationFields}{%
10657 name={\glsxtrlongshortdescname},
10658 sort={\glsxtrlongshortdescsort},%
10659 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%

```

```

10660 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10661 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10662 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10663 }%
10664 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10665   \csdef{glxtrpostlink\glscategorylabel}{%
10666     \glxtrifwasfirstuse
10667     {%
10668       \glxtrposthyphenshort{\glslabel}{\glsinsert}}%
10669     }%
10670     {%

```

Put the insertion into the post-link:

```

10671       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}}%
10672     }%
10673   }%
10674   \glshasattribute{\the\glslabeltok}{regular}%
10675   {%
10676     \glissetattribute{\the\glslabeltok}{regular}{false}%
10677   }%
10678   {}%
10679 }%
10680 }%
10681 {%
10682   \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10683 }

```

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.

```
10684 \newcommand*{\glxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10685 {%
```

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.)

```

10686   \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}}%
10687   \glsfirstabbrvhyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10688   \ifglxtrininsertinside\else{#4}\fi
10689   \glxtrfullsep{#1}%
10690   \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10691     \ifglxtrininsertinside\else{#4}\fi}%
10692 }%
10693 }

```

hen-long-hyphen Designed for use with the markwords attribute.

```
10694 \newabbreviationstyle{short-hyphen-long-hyphen}%
10695 {%
10696   \renewcommand*{\CustomAbbreviationFields}{%
10697     name={\glxtrshortlongname},
10698     sort={\the\glsshorttok},
10699     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10700       \protect\glxtrfullsep{\the\glslabeltok}%
10701       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10702     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10703       \protect\glxtrfullsep{\the\glslabeltok}%
10704       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10705     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}},%
10706     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10707 }
```

Unset the regular attribute if it has been set.

```
10707 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10708   \glshasattribute{\the\glslabeltok}{regular}%
10709   {%
10710     \glissetattribute{\the\glslabeltok}{regular}{false}%
10711   }%
10712 }%
10713 }%
10714 }%
10715 {%
10716   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10717   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10718   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10719   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10720   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
10721 }
```

The first use full form and the inline full form are the same for this style.

```
10721 \renewcommand*{\glxtrfullformat}[2]{%
10722   \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10723 }%
10724 \renewcommand*{\glxtrfullplformat}[2]{%
10725   \glxtrshorthyphenlong{##1}%
10726   {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10727 }%
10728 \renewcommand*{\Glsxtrfullformat}[2]{%
10729   \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10730 }%
10731 \renewcommand*{\Glsxtrfullplformat}[2]{%
10732   \glxtrshorthyphenlong{##1}%
10733   {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
10734 }%
10735 }
```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```
10736 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
10737 }
```

```

10737 {%
10738   \renewcommand*{\CustomAbbreviationFields}{%
10739     name={\glxtrshortlongdescname},
10740     sort={\glxtrshortlongdescsort},
10741     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10742       \protect\glxtrfullsep{\the\glslabeltok}%
10743       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10744     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10745       \protect\glxtrfullsep{\the\glslabeltok}%
10746       \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10747     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10748     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
10749   }%

```

Unset the regular attribute if it has been set.

```

10750   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10751     \glshasattribute{\the\glslabeltok}{regular}%
10752     {%
10753       \glissetattribute{\the\glslabeltok}{regular}{false}%
10754     }%
10755   }%
10756 }%
10757 }%
10758 {%
10759   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
10760 }

```

`\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.

```
10761 \newcommand*{\glxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```

10762 {%
10763   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}}%
10764   \glsfirstabbrvhyphenfont{#1}%
10765 }%
10766 }

```

`\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.



```

10767 \newcommand*{\glxtrposthyphenlong}[2]{%
10768 {%
10769   \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
10770   \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
10771   \glxtrfullsep{#1}%
10772   \glxtrparen
10773   {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglxtrininsertinside{#2}\fi}%
10774   \ifglxtrininsertinside\else{#2}\fi
10775 }%
10776 }%
10777 }

```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10778 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
10779 {%
10780   \renewcommand*{\CustomAbbreviationFields}{%
10781     name={\glxtrshortlongname},
10782     sort={\the\glsshorttok},
10783     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10784     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10785     plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}},%
10786     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10787   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10788     \csdef{glxtrpostlink\glscategorylabel}{%
10789       \glxtrifwasfirstuse
10790       {%
10791         \glxtrposthyphenlong{\glslabel}{\glsininsert}%
10792       }%
10793     }%

```

Put the insertion into the post-link:

```

10794       \glxtrposthyphenlong{\glslabel}{\glsininsert}%
10795     }%
10796   }%
10797   \glshasattribute{\the\glslabeltok}{regular}%
10798   {%
10799     \glissetattribute{\the\glslabeltok}{regular}{false}%
10800   }%
10801   {}%
10802 }%
10803 }%
10804 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10805 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10806 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
10807 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10808 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10809 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10810 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10811   \glsabbrvfont{\glsaccessshort{##1}}%
10812 }%
10813 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10814   \glsabbrvfont{\glsaccessshortpl{##1}}%
10815 }%
10816 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10817   \glsabbrvfont{\Glsaccessshort{##1}}%
10818 }%
10819 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10820   \glsabbrvfont{\Glsaccessshortpl{##1}}%
10821 }%

```

First use full form:

```

10822 \renewcommand*{\glxtrfullformat}[2]{%
10823   \glxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
10824 }%
10825 \renewcommand*{\glxtrfullplformat}[2]{%
10826   \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
10827 }%
10828 \renewcommand*{\Glsxtrfullformat}[2]{%
10829   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
10830 }%
10831 \renewcommand*{\Glsxtrfullplformat}[2]{%
10832   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
10833 }%

```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

10834 \renewcommand*{\glxtrinlinefullformat}[2]{%
10835   \glsfirstabbrvhyphenfont{\glsaccessshort{##1}}%
10836   \ifglxtrininsertinside{##2}\fi}%
10837 \ifglxtrininsertinside \else{##2}\fi
10838 }%
10839 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10840   \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}}%
10841   \ifglxtrininsertinside{##2}\fi}%
10842 \ifglxtrininsertinside \else{##2}\fi
10843 }%
10844 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10845   \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}}%
10846   \ifglxtrininsertinside{##2}\fi}%
10847 \ifglxtrininsertinside \else{##2}\fi
10848 }%
10849 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10850   \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}}%
10851   \ifglxtrininsertinside{##2}\fi}%
10852 \ifglxtrininsertinside \else{##2}\fi
10853 }%

```

10854 }

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

10855 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%

10856 {%

10857 \renewcommand\*{\CustomAbbreviationFields}{%

10858 name={\glstrshortlongdescname},

10859 sort={\glstrshortlongdescsort},%

10860 first={\protect\glfirstabbrvhyphenfont{\the\glsshorttok}},%

10861 firstplural={\protect\glfirstabbrvhyphenfont{\the\glsshortpltok}},%

10862 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%

10863 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%

10864 }%

10865 \renewcommand\*{\GlsXtrPostNewAbbreviation}{%

10866 \csdef{glstrpostlink\glscategorylabel}{%

10867 \glstrifwasfirstuse

10868 {%

10869 \glstrposthyphenlong{\glslabel}{\glsinsert}%

10870 }%

10871 {%

Put the insertion into the post-link:

10872 \glstrposthyphensubsequent{\glslabel}{\glsinsert}%

10873 }%

10874 }%

10875 \glshasattribute{\the\glslabeltok}{regular}%

10876 {%

10877 \glissetattribute{\the\glslabeltok}{regular}{false}%

10878 }%

10879 {}%

10880 }%

10881 }%

10882 {%

10883 \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%

10884 }

### 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

10885 \newcommand\*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%

stabbrvonlyfont

10886 \newcommand\*{\glfirstabbrvonlyfont}{\glsabbrvonlyfont}%

glslongonlyfont

10887 \newcommand\*{\glslongonlyfont}{\glslongdefaultfont}%

rstlongonlyfont

```
10888 \newcommand*{\glfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
10889 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtronlyname The default name format for this style.

```
10890 \newcommand*{\glsxtronlyname}{%
10891   \protect\glsabbrvonlyfont{\the\glsshorttok}%
10892 }
```

only-short-only

```
10893 \newabbreviationstyle{long-only-short-only}%
10894 {%
10895   \renewcommand*{\CustomAbbreviationFields}{%
10896     name={\glsxtronlyname},
10897     sort={\the\glsshorttok},
10898     first={\protect\glfirstlongonlyfont{\the\glslongtok}},%
10899     firstplural={\protect\glfirstlongonlyfont{\the\glslongpltok}},%
10900     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
10901     description={\protect\glslongonlyfont{\the\glslongtok}}}%
10902 }
```

Unset the regular attribute if it has been set.

```
10902 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10903   \glshasattribute{\the\glslabeltok}{regular}%
10904   {%
10905     \glissetattribute{\the\glslabeltok}{regular}{false}%
10906   }%
10907   {}%
10908 }%
10909 }%
10910 {%
10911   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtronlysuffix}%
10912   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
10913   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
10914   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
10915   \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%
10916 }
```

The first use full form doesn't show the short form.

```
10916 \renewcommand*{\glsxtrfullformat}[2]{%
10917   \glfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10918   \ifglsxtrinsertinside\else##2\fi
10919 }%
10920 \renewcommand*{\glsxtrfullplformat}[2]{%
10921   \glfirstlongonlyfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
10922   \ifglsxtrinsertinside\else##2\fi
10923 }%
10924 \renewcommand*{\Glsxtrfullformat}[2]{%
10925   \glfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
10926   \ifglsxtrinsertinside\else##2\fi
10927 }
```

```

10925 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10926 \ifglxtrinsertinside\else##2\fi
10927 }%
10928 \renewcommand*{\Glsxtrfullplformat}[2]{%
10929 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10930 \ifglxtrinsertinside\else##2\fi
10931 }%

```

The inline full form does show the short form.

```

10932 \renewcommand*{\glxtrinlinefullformat}[2]{%
10933 \glsfirstlongonlyfont{\glaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10934 \ifglxtrinsertinside\else##2\fi
10935 \glxtrfullsep{##1}%
10936 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshort{##1}}}%
10937 }%
10938 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10939 \glsfirstlongonlyfont{\glaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10940 \ifglxtrinsertinside\else##2\fi
10941 \glxtrfullsep{##1}%
10942 \glxtrparen{\protect\glfirstabbrvonlyfont{\glaccessshortpl{##1}}}%
10943 }%
10944 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10945 \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10946 \ifglxtrinsertinside\else##2\fi
10947 \glxtrfullsep{##1}%
10948 \glxtrparen{\protect\glfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10949 }%
10950 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10951 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10952 \ifglxtrinsertinside\else##2\fi
10953 \glxtrfullsep{##1}%
10954 \glxtrparen{\protect\glfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10955 }%
10956 }

```

xtonlydescsort

```

10957 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtonlydescname

```

10958 \newcommand*{\glxtronlydescname}{%
10959 \protect\glslongfont{\the\glslongtok}%
10960 }

```

short-only-desc

```

10961 \newabbreviationstyle{long-only-short-only-desc}%
10962 {%
10963 \renewcommand*{\CustomAbbreviationFields}{%
10964 name={\glxtronlydescname},
10965 sort={\glxtronlydescsort},%

```

```

10966   first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10967   firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10968   text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
10969   plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}}%
10970 }%

  Unset the regular attribute if it has been set.
10971 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10972   \glshasattribute{\the\glslabeltok}{regular}}%
10973   {%
10974     \glssetattribute{\the\glslabeltok}{regular}{false}}%
10975   }%
10976   {}%
10977 }%
10978 }%
10979 {%
10980   \GlsXtrUseAbbrStyleFmts{long-only-short-only}}%
10981 }

```

## 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:

```

10982 \let\@glsxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```
10983 \renewcommand*{\markright}[1]{%
10984   \glxtrmarkhook
10985   \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
10986   \glxtrrestoremarkhook
10987 }
```

\markboth Save original definition:

```
10988 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
10989 \renewcommand*{\markboth}[2]{%
10990   \glxtrmarkhook
10991   \@glxtr@org@markboth
10992   {\@glxtrinmark#1\@glxtrnotinmark}%
10993   {\@glxtrinmark#2\@glxtrnotinmark}%
10994   \glxtrrestoremarkhook
10995 }
```

Also do this for \@starttoc

\@starttoc Save original definition:

```
10996 \let\@glxtr@org@@starttoc\@starttoc
```

Redefine:

```
10997 \renewcommand*{\@starttoc}[1]{%
10998   \glxtrmarkhook
10999   \@glxtrinmark
11000   \@glxtr@org@@starttoc{#1}%
11001   \@glxtrnotinmark
11002   \glxtrrestoremarkhook
11003 }
```

If this causes a problem provide a simple way of switching back to the original definitions:

glxtrRevertMarks

```
11004 \newcommand*{\glxtrRevertMarks}{%
11005   \let\markright\@glxtr@org@markright
11006   \let\markboth\@glxtr@org@markboth
11007   \let\@starttoc\@glxtr@org@@starttoc
11008 }
```

glxtrRevertTocMarks Just restores \@starttoc.

```
11009 \newcommand*{\glxtrRevertTocMarks}{%
11010   \let\@starttoc\@glxtr@org@@starttoc
11011 }
```

\glxtrifinmark

```
11012 \newcommand*{\glxtrifinmark}[2]{#2}
```

\@glsxtrinmark

```
11013 \newrobustcmd*{\@glsxtrinmark}{%
11014   \let\glsxtrifinmark\@firstoftwo
11015 }
```

glsxtrnotinmark

```
11016 \newrobustcmd*{\@glsxtrnotinmark}{%
11017   \let\glsxtrifinmark\@secondoftwo
11018 }
```

eorpdforheading

```
11019 \ifdef\texorpdfstring
11020 {
11021   \newcommand*{\glsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
11022 }
11023 {
11024   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
11025 }
```

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
11026 \newcommand*{\glsxtrmarkhook}{%
```

Save current definitions:

```
11027 \let\@glsxtr@org@MakeUppercase\MakeUppercase
11028 \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
11029 \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
11030 \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
11031 \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
11032 \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
11033 \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
11034 \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
11035 \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
11036 \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
11037 \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
11038 \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
11039 \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
11040 \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
11041 \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
11042 \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
11043 \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
11044 \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
11045 \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
11046 \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
11047 \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
11048 \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
11049 \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
11050 \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```



## New definitions

```

11051 \let\glsxrtrifinmark\@firstoftwo
11052 \let\MakeUppercase\MakeTextUppercase
11053 \let\glsxtrtitleorpdforheading\@thirdofthree
11054 \let\glsxtrtitleshort\glsxtrheadshort
11055 \let\glsxtrtitleshortpl\glsxtrheadshortpl
11056 \let\Glsxtrtitleshort\Glsxtrheadshort
11057 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
11058 \let\glsxtrtitlename\glsxtrheadname
11059 \let\Glsxtrtitlename\Glsxtrheadname
11060 \let\glsxtrtitletext\glsxtrheadtext
11061 \let\Glsxtrtitletext\Glsxtrheadtext
11062 \let\glsxtrtitleplural\glsxtrheadplural
11063 \let\Glsxtrtitleplural\Glsxtrheadplural
11064 \let\glsxtrtitlefirst\glsxtrheadfirst
11065 \let\Glsxtrtitlefirst\Glsxtrheadfirst
11066 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
11067 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
11068 \let\glsxtrtitlelong\glsxtrheadlong
11069 \let\glsxtrtitlelongpl\glsxtrheadlongpl
11070 \let\Glsxtrtitlelong\Glsxtrheadlong
11071 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
11072 \let\glsxtrtitlefull\glsxtrheadfull
11073 \let\glsxtrtitlefullpl\glsxtrheadfullpl
11074 \let\Glsxtrtitlefull\Glsxtrheadfull
11075 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
11076 }

```

`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.)

```

11077 \newcommand*{\glsxtrrestoremarkhook}{%
11078 \let\glsxrtrifinmark\@secondoftwo
11079 \let\MakeUppercase\@glsxtr@org@MakeUppercase
11080 \let\glsxtrtitleorpdforheading\@glsxtr@org@glsxtrtitleorpdforheading
11081 \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
11082 \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
11083 \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
11084 \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
11085 \let\glsxtrtitlename\@glsxtr@org@glsxtrtitlename
11086 \let\Glsxtrtitlename\@glsxtr@org@Glsxtrtitlename
11087 \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
11088 \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
11089 \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
11090 \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
11091 \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
11092 \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
11093 \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural

```

```

11094 \let\Glsxtrtitlefirstplural\@glxstr@org@Glsxtrtitlefirstplural
11095 \let\glxstrtitlelong\@glxstr@org@glxstrtitlelong
11096 \let\glxstrtitlelongpl\@glxstr@org@glxstrtitlelongpl
11097 \let\Glsxtrtitlelong\@glxstr@org@Glsxtrtitlelong
11098 \let\Glsxtrtitlelongpl\@glxstr@org@Glsxtrtitlelongpl
11099 \let\glxstrtitlefull\@glxstr@org@glxstrtitlefull
11100 \let\glxstrtitlefullpl\@glxstr@org@glxstrtitlefullpl
11101 \let\Glsxtrtitlefull\@glxstr@org@Glsxtrtitlefull
11102 \let\Glsxtrtitlefullpl\@glxstr@org@Glsxtrtitlefullpl
11103 }

```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

**glxstrheadshort** Command used to display short form in the page header.

```

11104 \newcommand*{\glxstrheadshort}[1]{%
11105   \protect\NoCaseChange
11106   {%
11107     \gl@ifattribute{#1}{headuc}{true}%
11108     {%
11109       \GLSxtrshort[noindex,hyper=false]{#1}[]%
11110     }%
11111     {%
11112       \glxstrshort[noindex,hyper=false]{#1}[]%
11113     }%
11114   }%
11115 }

```

**lsxtrtitleshort** Command to display short form of abbreviation in section title and table of contents.

```

11116 \newrobustcmd*{\lsxtrtitleshort}[1]{%
11117   \glxstrshort[noindex,hyper=false]{#1}[]%
11118 }

```

**glxstrheadshortpl** Command used to display plural short form in the page header. If you want the text converted to upper case, this needs to be redefined to use \GLSxtrshortpl instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```

11119 \newcommand*{\glxstrheadshortpl}[1]{%
11120   \protect\NoCaseChange
11121   {%
11122     \gl@ifattribute{#1}{headuc}{true}%
11123     {%
11124       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
11125     }%
11126     {%
11127       \glxstrshortpl[noindex,hyper=false]{#1}[]%
11128     }%
11129   }%
11130 }

```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```
11131 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11132   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11133 }
```

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
11134 \newcommand*{\Glsxtrheadshort}[1]{%
11135   \protect\NoCaseChange
11136   {%
11137     \glsifattribute{#1}{headuc}{true}%
11138     {%
11139       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11140     }%
11141     {%
11142       \Glsxtrshort[noindex,hyper=false]{#1}[]%
11143     }%
11144   }%
11145 }
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11146 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
11147   \Glsxtrshort[noindex,hyper=false]{#1}[]%
11148 }
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
11149 \newcommand*{\Glsxtrheadshortpl}[1]{%
11150   \protect\NoCaseChange
11151   {%
11152     \glsifattribute{#1}{headuc}{true}%
11153     {%
11154       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11155     }%
11156     {%
11157       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11158     }%
11159   }%
11160 }
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11161 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
11162   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
11163 }
```

`\glxtrheadname` As above but for the name value.

```

11164 \newcommand*{\glxtrheadname}[1]{%
11165   \protect\NoCaseChange
11166   {%
11167     \gl@ifattribute{#1}{headuc}{true}%
11168     {%
11169       \GLSname[noindex,hyper=false]{#1}[]%
11170     }%
11171     {%
11172       \GLSname[noindex,hyper=false]{#1}[]%
11173     }%
11174   }%
11175 }

```

**glxtrtitlename** Command to display name value in section title and table of contents.

```

11176 \newrobustcmd*{\glxtrtitlename}[1]{%
11177   \GLSname[noindex,hyper=false]{#1}[]%
11178 }

```

**\Glsxtrheadname** First letter converted to upper case

```

11179 \newcommand*{\Glsxtrheadname}[1]{%
11180   \protect\NoCaseChange
11181   {%
11182     \gl@ifattribute{#1}{headuc}{true}%
11183     {%
11184       \GLSname[noindex,hyper=false]{#1}[]%
11185     }%
11186     {%
11187       \GLSname[noindex,hyper=false]{#1}[]%
11188     }%
11189   }%
11190 }

```

**Glsxtrtitlename** Command to display name value in section title and table of contents with the first letter changed to upper case.

```

11191 %\changes{1.21}{2017-11-03}{new}
11192 \newrobustcmd*{\Glsxtrtitlename}[1]{%
11193   \GLSname[noindex,hyper=false]{#1}[]%
11194 }

```

**\glxtrheadtext** As above but for the text value.

```

11195 \newcommand*{\glxtrheadtext}[1]{%
11196   \protect\NoCaseChange
11197   {%
11198     \gl@ifattribute{#1}{headuc}{true}%
11199     {%
11200       \GLStext[noindex,hyper=false]{#1}[]%
11201     }%
11202     {%
11203       \GLStext[noindex,hyper=false]{#1}[]%

```

```

11204 }%
11205 }%
11206 }

```

**glsxtrtitletext** Command to display text value in section title and table of contents.

```

11207 \newrobustcmd*{\glsxtrtitletext}[1]{%
11208   \glstext[noindex,hyper=false]{#1}[]%
11209 }

```

**\Glsxtrheadtext** First letter converted to upper case

```

11210 \newcommand*{\Glsxtrheadtext}[1]{%
11211   \protect\NoCaseChange
11212   {%
11213     \glsifattribute{#1}{headuc}{true}%
11214     {%
11215       \GLStext[noindex,hyper=false]{#1}[]%
11216     }%
11217     {%
11218       \Glstext[noindex,hyper=false]{#1}[]%
11219     }%
11220   }%
11221 }

```

**Glsxtrtitletext** Command to display text value in section title and table of contents with the first letter changed to upper case.

```

11222 \newrobustcmd*{\Glsxtrtitletext}[1]{%
11223   \Glstext[noindex,hyper=false]{#1}[]%
11224 }

```

**lsxtrheadplural** As above but for the plural value.

```

11225 \newcommand*{\lsxtrheadplural}[1]{%
11226   \protect\NoCaseChange
11227   {%
11228     \glsifattribute{#1}{headuc}{true}%
11229     {%
11230       \GLSplural[noindex,hyper=false]{#1}[]%
11231     }%
11232     {%
11233       \glsplural[noindex,hyper=false]{#1}[]%
11234     }%
11235   }%
11236 }

```

**sxtrtitleplural** Command to display plural value in section title and table of contents.

```

11237 \newrobustcmd*{\sxtrtitleplural}[1]{%
11238   \glsplural[noindex,hyper=false]{#1}[]%
11239 }

```

**lsxtrheadplural** Convert first letter to upper case.

```
11240 \newcommand*{\Glsxtrheadplural}[1]{%
11241   \protect\NoCaseChange
11242   {%
11243     \glsifattribute{#1}{headuc}{true}%
11244     {%
11245       \GLSplural[noindex,hyper=false]{#1}[]%
11246     }%
11247     {%
11248       \Glsplural[noindex,hyper=false]{#1}[]%
11249     }%
11250   }%
11251 }
```

**sxtrtitleplural** Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
11252 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
11253   \Glsplural[noindex,hyper=false]{#1}[]%
11254 }
```

**glsxtrheadfirst** As above but for the first value.

```
11255 \newcommand*{\glsxtrheadfirst}[1]{%
11256   \protect\NoCaseChange
11257   {%
11258     \glsifattribute{#1}{headuc}{true}%
11259     {%
11260       \GLSfirst[noindex,hyper=false]{#1}[]%
11261     }%
11262     {%
11263       \glsfirst[noindex,hyper=false]{#1}[]%
11264     }%
11265   }%
11266 }
```

**lsxtrtitlefirst** Command to display first value in section title and table of contents.

```
11267 \newrobustcmd*{\glsxtrtitlefirst}[1]{%
11268   \glsfirst[noindex,hyper=false]{#1}[]%
11269 }
```

**Glsxtrheadfirst** First letter converted to upper case

```
11270 \newcommand*{\Glsxtrheadfirst}[1]{%
11271   \protect\NoCaseChange
11272   {%
11273     \glsifattribute{#1}{headuc}{true}%
11274     {%
11275       \GLSfirst[noindex,hyper=false]{#1}[]%
11276     }%
11277     {%
```

```

11278     \Glsfirst[noindex,hyper=false]{#1}[]%
11279 }%
11280 }%
11281 }

```

`\lsxtrtitlefirst` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11282 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
11283   \Glsfirst[noindex,hyper=false]{#1}[]%
11284 }

```

`\headfirstplural` As above but for the firstplural value.

```

11285 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11286   \protect\NoCaseChange
11287   {%
11288     \glsifattribute{#1}{headuc}{true}%
11289     {%
11290       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11291     }%
11292     {%
11293       \glsfirstplural[noindex,hyper=false]{#1}[]%
11294     }%
11295   }%
11296 }

```

`\itlefirstplural` Command to display firstplural value in section title and table of contents.

```

11297 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11298   \glsfirstplural[noindex,hyper=false]{#1}[]%
11299 }

```

`\headfirstplural` First letter converted to upper case

```

11300 \newcommand*{\Glsxtrheadfirstplural}[1]{%
11301   \protect\NoCaseChange
11302   {%
11303     \glsifattribute{#1}{headuc}{true}%
11304     {%
11305       \GLSfirstplural[noindex,hyper=false]{#1}[]%
11306     }%
11307     {%
11308       \Glsfirstplural[noindex,hyper=false]{#1}[]%
11309     }%
11310   }%
11311 }

```

`\itlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

11312 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
11313   \Glsfirstplural[noindex,hyper=false]{#1}[]%
11314 }

```

`\glxstrheadlong` Command used to display long form in the page header.

```
11315 \newcommand*{\glxstrheadlong}[1]{%
11316   \protect\NoCaseChange
11317   {%
11318     \gl@ifattribute{#1}{headuc}{true}%
11319     {%
11320       \GLSxtrlong[noindex,hyper=false]{#1}[]%
11321     }%
11322     {%
11323       \glxstrlong[noindex,hyper=false]{#1}[]%
11324     }%
11325   }%
11326 }
```

`glxstrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
11327 \newrobustcmd*{\glxstrtitlelong}[1]{%
11328   \glxstrlong[noindex,hyper=false]{#1}[]%
11329 }
```

`lsxtrheadlongpl` Command used to display plural long form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrlongpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
11330 \newcommand*{\glxtrheadlongpl}[1]{%
11331   \protect\NoCaseChange
11332   {%
11333     \gl@ifattribute{#1}{headuc}{true}%
11334     {%
11335       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11336     }%
11337     {%
11338       \glxtrlongpl[noindex,hyper=false]{#1}[]%
11339     }%
11340   }%
11341 }
```

`sxstrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
11342 \newrobustcmd*{\glxstrtitlelongpl}[1]{%
11343   \glxtrlongpl[noindex,hyper=false]{#1}[]%
11344 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```
11345 \newcommand*{\Glsxtrheadlong}[1]{%
11346   \protect\NoCaseChange
11347   {%
11348     \gl@ifattribute{#1}{headuc}{true}%
11349     {%
11350       \GLSxtrlong[noindex,hyper=false]{#1}[]%
```



```

11351 }%
11352 {%
11353     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11354 }%
11355 }%
11356 }

```

**Glsxtrtitlelong** Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11357 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
11358     \Glsxtrlong[noindex,hyper=false]{#1}[]%
11359 }

```

**lgsxtrheadlongpl** Command used to display plural long form in the page header with the first letter converted to upper case.

```

11360 \newcommand*{\Glsxtrheadlongpl}[1]{%
11361     \protect\NoCaseChange
11362     {%
11363         \glsifattribute{#1}{headuc}{true}%
11364         {%
11365             \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
11366         }%
11367         {%
11368             \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11369         }%
11370     }%
11371 }

```

**sxtrtitlelongpl** Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11372 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
11373     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
11374 }

```

**\glsxtrheadfull** Command used to display full form in the page header.

```

11375 \newcommand*{\glsxtrheadfull}[1]{%
11376     \protect\NoCaseChange
11377     {%
11378         \glsifattribute{#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.

```
11387 \newrobustcmd*{\glsxtrtitlefull}[1]{%
11388   \glsxtrfull[noindex,hyper=false]{#1}[]%
11389 }
```

`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.

```
11390 \newcommand*{\glsxtrheadfullpl}[1]{%
11391   \protect\NoCaseChange
11392   {%
11393     \glsifattribute{#1}{headuc}{true}%
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.

```
11402 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%
11403   \glsxtrfullpl[noindex,hyper=false]{#1}[]%
11404 }
```

`\GLsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```
11405 \newcommand*{\GLsxtrheadfull}[1]{%
11406   \protect\NoCaseChange
11407   {%
11408     \glsifattribute{#1}{headuc}{true}%
11409     {%
11410       \GLSxtrfull[noindex,hyper=false]{#1}[]%
11411     }%
11412     {%
11413       \GLsxtrfull[noindex,hyper=false]{#1}[]%
11414     }%
11415   }%
11416 }
```

`Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
11417 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
11418   \GLsxtrfull[noindex,hyper=false]{#1}[]%
11419 }
```

`\lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

11420 \newcommand*{\Glsxtrheadfullpl}[1]{%
11421   \protect\NoCaseChange
11422   {%
11423     \glsifattribute{#1}{headuc}{true}%
11424     {%
11425       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11426     }%
11427     {%
11428       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11429     }%
11430   }%
11431 }

```

`\sxttrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

11432 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
11433   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
11434 }

```

`\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.

```

11435 \ifdef\texorpdfstring
11436 {
11437   \newcommand*{\glsfmtshort}[1]{%
11438     \texorpdfstring
11439     {\glsxtrtitleshort{#1}}%
11440     {\glsentryshort{#1}}%
11441   }
11442 }
11443 {
11444   \newcommand*{\glsfmtshort}[1]{%
11445     \glsxtrtitleshort{#1}}
11446 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

11447 \ifdef\texorpdfstring
11448 {
11449   \newcommand*{\glsfmtshortpl}[1]{%
11450     \texorpdfstring
11451     {\glsxtrtitleshortpl{#1}}%
11452     {\glsentryshortpl{#1}}%
11453   }
11454 }
11455 {
11456   \newcommand*{\glsfmtshortpl}[1]{%

```

```

11457 \glstrtitleshortpl{#1}}
11458 }

```

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).

```

11459 \ifdef\texorpdfstring
11460 {
11461   \newcommand*\Glsfmtshort[1]{%
11462     \texorpdfstring
11463       {\Glsxtrtitleshort{#1}}%
11464       {\glsentryshort{#1}}%
11465   }
11466 }
11467 {
11468   \newcommand*\Glsfmtshort[1]{%
11469     \Glsxtrtitleshort{#1}}
11470 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

11471 \ifdef\texorpdfstring
11472 {
11473   \newcommand*\Glsfmtshortpl[1]{%
11474     \texorpdfstring
11475       {\Glsxtrtitleshortpl{#1}}%
11476       {\glsentryshortpl{#1}}%
11477   }
11478 }
11479 {
11480   \newcommand*\Glsfmtshortpl[1]{%
11481     \Glsxtrtitleshortpl{#1}}
11482 }

```

`\glsfmtname` As above but for the name value.

```

11483 \ifdef\texorpdfstring
11484 {
11485   \newcommand*\glsfmtname[1]{%
11486     \texorpdfstring
11487       {\glsxtrtitlename{#1}}%
11488       {\glsentryname{#1}}%
11489   }
11490 }
11491 {
11492   \newcommand*\glsfmtname[1]{%
11493     \glsxtrtitlename{#1}}
11494 }

```

`\Glsfmtname` First letter converted to upper case.

```

11495 \ifdef\texorpdfstring
11496 {
11497   \newcommand*{\Glsfmtname}[1]{%
11498     \texorpdfstring
11499     {\Glsxtrtitlename{#1}}%
11500     {\glsentryname{#1}}%
11501   }
11502 }
11503 {
11504   \newcommand*{\Glsfmtname}[1]{%
11505     \Glsxtrtitlename{#1}}
11506 }

```

`\glsfmttext` As above but for the text value.

```

11507 \ifdef\texorpdfstring
11508 {
11509   \newcommand*{\glsfmttext}[1]{%
11510     \texorpdfstring
11511     {\glsxtrtitletext{#1}}%
11512     {\glsentrytext{#1}}%
11513   }
11514 }
11515 {
11516   \newcommand*{\glsfmttext}[1]{%
11517     \glsxtrtitletext{#1}}
11518 }

```

`\Glsfmttext` First letter converted to upper case.

```

11519 \ifdef\texorpdfstring
11520 {
11521   \newcommand*{\Glsfmttext}[1]{%
11522     \texorpdfstring
11523     {\Glsxtrtitletext{#1}}%
11524     {\glsentrytext{#1}}%
11525   }
11526 }
11527 {
11528   \newcommand*{\Glsfmttext}[1]{%
11529     \Glsxtrtitletext{#1}}
11530 }

```

`\glsfmtplural` As above but for the plural value.

```

11531 \ifdef\texorpdfstring
11532 {
11533   \newcommand*{\glsfmtplural}[1]{%
11534     \texorpdfstring
11535     {\glsxtrtitleplural{#1}}%
11536     {\glsentryplural{#1}}%
11537   }

```

```

11538 }
11539 {
11540   \newcommand*{\glsfmtplural}[1]{%
11541     \glsxtrtitleplural{#1}}
11542 }

```

`\Glsfmtplural` First letter converted to upper case.

```

11543 \ifdef\texorpdfstring
11544 {
11545   \newcommand*{\Glsfmtplural}[1]{%
11546     \texorpdfstring
11547     {\Glsxtrtitleplural{#1}}%
11548     {\glsentryplural{#1}}%
11549   }
11550 }
11551 {
11552   \newcommand*{\Glsfmtplural}[1]{%
11553     \Glsxtrtitleplural{#1}}
11554 }

```

`\glsfmtfirst` As above but for the first value.

```

11555 \ifdef\texorpdfstring
11556 {
11557   \newcommand*{\glsfmtfirst}[1]{%
11558     \texorpdfstring
11559     {\glsxtrtitlefirst{#1}}%
11560     {\glsentryfirst{#1}}%
11561   }
11562 }
11563 {
11564   \newcommand*{\glsfmtfirst}[1]{%
11565     \glsxtrtitlefirst{#1}}
11566 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

11567 \ifdef\texorpdfstring
11568 {
11569   \newcommand*{\Glsfmtfirst}[1]{%
11570     \texorpdfstring
11571     {\Glsxtrtitlefirst{#1}}%
11572     {\glsentryfirst{#1}}%
11573   }
11574 }
11575 {
11576   \newcommand*{\Glsfmtfirst}[1]{%
11577     \Glsxtrtitlefirst{#1}}
11578 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

11579 \ifdef\texorpdfstring
11580 {
11581   \newcommand*\glsfmtfirstpl}[1]{%
11582     \texorpdfstring
11583     {\glsxtrtitlefirstplural{#1}}%
11584     {\glsentryfirstplural{#1}}%
11585   }
11586 }
11587 {
11588   \newcommand*\glsfmtfirstpl}[1]{%
11589     \glsxtrtitlefirstplural{#1}}
11590 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

11591 \ifdef\texorpdfstring
11592 {
11593   \newcommand*\Glsfmtfirstpl}[1]{%
11594     \texorpdfstring
11595     {\Glsxtrtitlefirstplural{#1}}%
11596     {\glsentryfirstplural{#1}}%
11597   }
11598 }
11599 {
11600   \newcommand*\Glsfmtfirstpl}[1]{%
11601     \Glsxtrtitlefirstplural{#1}}
11602 }

```

`\glsfmtlong` As above but for the long value.

```

11603 \ifdef\texorpdfstring
11604 {
11605   \newcommand*\glsfmtlong}[1]{%
11606     \texorpdfstring
11607     {\glsxtrtitlelong{#1}}%
11608     {\glsentrylong{#1}}%
11609   }
11610 }
11611 {
11612   \newcommand*\glsfmtlong}[1]{%
11613     \glsxtrtitlelong{#1}}
11614 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11615 \ifdef\texorpdfstring
11616 {
11617   \newcommand*\Glsfmtlong}[1]{%
11618     \texorpdfstring
11619     {\Glsxtrtitlelong{#1}}%
11620     {\glsentrylong{#1}}%
11621   }

```

```

11622 }
11623 {
11624   \newcommand*{\Glsfmtlong}[1]{%
11625     \Glsxtrtitlelong{#1}}
11626 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

11627 \ifdef\texorpdfstring
11628 {
11629   \newcommand*{\glsfmtlongpl}[1]{%
11630     \texorpdfstring
11631     {\Glsxtrtitlelongpl{#1}}%
11632     {\Glsentrylongpl{#1}}%
11633   }
11634 }
11635 {
11636   \newcommand*{\glsfmtlongpl}[1]{%
11637     \Glsxtrtitlelongpl{#1}}
11638 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

11639 \ifdef\texorpdfstring
11640 {
11641   \newcommand*{\Glsfmtlongpl}[1]{%
11642     \texorpdfstring
11643     {\Glsxtrtitlelongpl{#1}}%
11644     {\Glsentrylongpl{#1}}%
11645   }
11646 }
11647 {
11648   \newcommand*{\Glsfmtlongpl}[1]{%
11649     \Glsxtrtitlelongpl{#1}}
11650 }

```

`\glsfmtfull` In-line full format.

```

11651 \ifdef\texorpdfstring
11652 {
11653   \newcommand*{\glsfmtfull}[1]{%
11654     \texorpdfstring
11655     {\Glsxtrtitlefull{#1}}%
11656     {\Glsxtrinlinefullformat{#1}{}}%
11657   }
11658 }
11659 {
11660   \newcommand*{\glsfmtfull}[1]{%
11661     \Glsxtrtitlefull{#1}}
11662 }

```

`\Glsfmtfull` First letter converted to upper case.



```

11663 \ifdef\txorpdfstring
11664 {
11665   \newcommand*{\Glsfmtfull}[1]{%
11666     \txorpdfstring
11667     {\Glsxtrtitlefull{#1}}%
11668     {\Glsxtrinlinefullformat{#1}{}}%
11669   }
11670 }
11671 {
11672   \newcommand*{\Glsfmtfull}[1]{%
11673     \Glsxtrtitlefull{#1}}
11674 }

```

`\glsfmtfullpl` In-line full plural format.

```

11675 \ifdef\txorpdfstring
11676 {
11677   \newcommand*{\glsfmtfullpl}[1]{%
11678     \txorpdfstring
11679     {\glsxtrtitlefullpl{#1}}%
11680     {\glsxtrinlinefullplformat{#1}{}}%
11681   }
11682 }
11683 {
11684   \newcommand*{\glsfmtfullpl}[1]{%
11685     \glsxtrtitlefullpl{#1}}
11686 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

11687 \ifdef\txorpdfstring
11688 {
11689   \newcommand*{\Glsfmtfullpl}[1]{%
11690     \txorpdfstring
11691     {\Glsxtrtitlefullpl{#1}}%
11692     {\Glsxtrinlinefullplformat{#1}{}}%
11693   }
11694 }
11695 {
11696   \newcommand*{\Glsfmtfullpl}[1]{%
11697     \Glsxtrtitlefullpl{#1}}
11698 }

```

## 1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

11699 \newcommand*{\RequireGlossariesExtraLang}[1]{%

```

```

11700 \ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11701 }

```

sariesExtraLang

```

11702 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11703   \ProvidesFile{glossariesxtr-#1.ldf}%
11704 }

```

Load any required language modules that are available. This doesn't generate any warning if none are found, since they're not essential. (The only command that really needs defining for the document is `\abbreviationsname`, which can simply be redefined. However, with `bib2gls` it might be useful to provide custom rules for a particular locale.)

`xtr@loaddialect` The dialect label should be stored in `\this@dialect` before using this command.

```

11705 \newcommand{\glxtr@loaddialect}{%
11706   \IfTrackedLanguageFileExists{\this@dialect}%
11707   {glossariesxtr-}% prefix
11708   {.ldf}%
11709   {%
11710     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
11711   }%
11712   {}% not found

```

If `glossaries-extra-bib2gls` has been loaded, `\@glxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```

11713 \@glxtrdialecthook
11714 }

```

```

11715 \ifpackageloaded{tracklang}
11716 {%
11717   \AnyTrackedLanguages
11718   {%
11719     \ForEachTrackedDialect{\this@dialect}{\glxtr@loaddialect}%
11720   }%
11721   {}%
11722 }
11723 {}

```

Load `glossaries-extra-stylemods` if required.

```

11724 \@glxtr@redefstyles

```

and set the style:

```

11725 \@glxtr@do@style

```

## 1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the `record` option.

```

11726 \NeedsTeXFormat{LaTeX2e}
11727 \ProvidesPackage{glossaries-extra-bib2gls}[2018/07/29 v1.34 (NLCT)]

```

These are some convenient macros for use with custom rules.

`\glshex`

```

11728 \newcommand*{\glshex}{\string\u}

```

`\glscapturedgroup`

```

11729 \newcommand*{\glscapturedgroup}{\string\$}

```

`\GlsXtrIfHasNonZeroChildCount` For use with bib2gls's save-child-count resource option.

```

11730 \newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
11731   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
11732 }

```

`\GlsXtrProvideCommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```

11733 \newcommand*{\GlsXtrProvideCommand}{\providecommand}

```

`\GlsXtrWrglossaryLocation` For use with indexcounter and bib2gls.

```

11734 \newcommand*{\GlsXtrWrglossaryLocation}[2]{#1}

```

`\GlsXtrIndexCounterLink`

```
\GlsXtrIndexCounterLink{<text>}{<label>}
```

For use with indexcounter and bib2gls.

```

11735 \ifdef\hyperref
11736 {%
11737   \newcommand*{\GlsXtrIndexCounterLink}[2]{%
11738     \GlsXtrIfHasField{indexcounter}{#2}%
11739     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
11740     {#1}%
11741   }
11742 }
11743 {
11744   \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
11745 }

```

`\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.

```

11746 \newcommand*{\GlsXtrDualField}{dual}

```

sXtrDualBackLink

<code>\GlsXtrDualBackLink{&lt;text&gt;}{&lt;label&gt;}</code>
---

Adds a hyperlink to the dual entry.

```
11747 \newcommand*{\GlsXtrDualBackLink}[2]{%
11748   \glstrifhasfield{\GlsXtrDualField}{#2}%
11749   {\glshyperlink[#1]{\glscurrentfieldvalue}}%
11750   {#2}%
11751 }
```

TeXEntryAliases

Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```
11752 \newcommand*{\GlsXtrBibTeXEntryAliases}{%
11753   article=bibtexentry,
11754   book=bibtexentry,
11755   booklet=bibtexentry,
11756   conference=bibtexentry,
11757   inbook=bibtexentry,
11758   incollection=bibtexentry,
11759   inproceedings=bibtexentry,
11760   manual=bibtexentry,
11761   mastersthesis=bibtexentry,
11762   misc=bibtexentry,
11763   phdthesis=bibtexentry,
11764   proceedings=bibtexentry,
11765   techreport=bibtexentry,
11766   unpublished=bibtexentry
11767 }
```

ideBibTeXFields

Convenient shortcut to define the standard BibTeX fields.

```
11768 \newcommand*{\GlsXtrProvideBibTeXFields}{%
11769   \glsaddstoragekey{address}{\glstrbibaddress}%
11770   \glsaddstoragekey{author}{\glstrbibauthor}%
11771   \glsaddstoragekey{booktitle}{\glstrbibbooktitle}%
11772   \glsaddstoragekey{chapter}{\glstrbibchapter}%
11773   \glsaddstoragekey{edition}{\glstrbibedition}%
11774   \glsaddstoragekey{howpublished}{\glstrbibhowpublished}%
11775   \glsaddstoragekey{institution}{\glstrbibinstitution}%
11776   \glsaddstoragekey{journal}{\glstrbibjournal}%
11777   \glsaddstoragekey{month}{\glstrbibmonth}%
11778   \glsaddstoragekey{note}{\glstrbibnote}%
11779   \glsaddstoragekey{number}{\glstrbibnumber}%
11780   \glsaddstoragekey{organization}{\glstrbiborganization}%
11781   \glsaddstoragekey{pages}{\glstrbibpages}%
11782   \glsaddstoragekey{publisher}{\glstrbibpublisher}%
11783   \glsaddstoragekey{school}{\glstrbibschooll}%
11784   \glsaddstoragekey{series}{\glstrbibseries}%
11785   \glsaddstoragekey{title}{\glstrbibtitle}%

```

```

11786 \glsaddstoragekey{bibtextype}{-}{\glsxtrbibtype}%
11787 \glsaddstoragekey{volume}{-}{\glsxtrbibvolume}%
11788 }

```

Provide missing Greek letters for use in maths mode. These commands are recognised by bib2gls and will be mapped to the Mathematical Greek Italic letters. This ensures that the Greek letters that have the same shape as Latin letters are kept with the other mathematical Greek letters for sorting purposes. The  $\LaTeX$  version of these commands (provided here) use an upright font for capitals and italic for lower case to provide a better match with the other Greek symbols provided by the kernel.

$\backslash$ Alpha

```
11789 \providecommand*{\Alpha}{\mathrm{A}}
```

$\backslash$ Beta

```
11790 \providecommand*{\Beta}{\mathrm{B}}
```

$\backslash$ Epsilon

```
11791 \providecommand*{\Epsilon}{\mathrm{E}}
```

$\backslash$ Zeta

```
11792 \providecommand*{\Zeta}{\mathrm{Z}}
```

$\backslash$ Eta

```
11793 \providecommand*{\Eta}{\mathrm{H}}
```

$\backslash$ Iota

```
11794 \providecommand*{\Iota}{\mathrm{I}}
```

$\backslash$ Kappa

```
11795 \providecommand*{\Kappa}{\mathrm{K}}
```

$\backslash$ Mu

```
11796 \providecommand*{\Mu}{\mathrm{M}}
```

$\backslash$ Nu

```
11797 \providecommand*{\Nu}{\mathrm{N}}
```

$\backslash$ Omicron

```
11798 \providecommand*{\Omicron}{\mathrm{O}}
```

$\backslash$ Rho

```
11799 \providecommand*{\Rho}{\mathrm{P}}
```

$\backslash$ Tau

```
11800 \providecommand*{\Tau}{\mathrm{T}}
```

```

\Chi
11801 \providecommand*\Chi{\mathrm{X}}

\Digamma
11802 \providecommand*\Digamma{\mathrm{F}}

\omicron
11803 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
11804 \@ifpackageloaded{upgreek}%
11805 {

\Upalpha
11806   \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
11807   \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
11808   \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
11809   \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
11810   \providecommand*\Upeta{\mathrm{H}}

\Upiota
11811   \providecommand*\Upiota{\mathrm{I}}

\Upkappa
11812   \providecommand*\Upkappa{\mathrm{K}}

\Upmu
11813   \providecommand*\Upmu{\mathrm{M}}

\Upnu
11814   \providecommand*\Upnu{\mathrm{N}}

\Upomicron
11815   \providecommand*\Upomicron{\mathrm{O}}

\Uprho
11816   \providecommand*\Uprho{\mathrm{P}}

```

`\Uptau`

```
11817 \providecommand*\Uptau{\mathrm{T}}
```

`\Upchi`

```
11818 \providecommand*\Upchi{\mathrm{X}}
```

`\upomicron`

```
11819 \providecommand*\upomicron{\mathrm{o}}
```

```
11820 }%
```

```
11821 {}% upgreek.sty not loaded
```

This package provides some basic rules, but it's not intended for complete coverage of all locales. The CLDR should provide the appropriate locale-sensitive rules. These macros are primarily to help construct custom rules to include, for example, Greek maths symbols mixed with Latin. For the full rule syntax, see the Java API for [RuleBaseCollator](#)

If you want to provide a rule-block for a particular locale to allow for customization within that locale, create a file called `glossariesxtr-⟨tag⟩.ldf` (where `⟨tag⟩` identifies the locale) and add similar commands. See the description of `\IfTrackedLanguageFileExists` in the `tracklang` manual for the allowed forms of `⟨tag⟩`. The simplest is to just use the root language label or ISO code. The file will then be automatically loaded by `glossaries-extra` if the document has support for that language.

When combining these blocks of rules, remember to separate them with the appropriate character. For example:

```
sort-rule={\glxstrcontrolrules
; \glxstrspacerules
; \glxstrnonprintablerules
; \glxstrcombiningdiacriticrules
, \glxstrhyphenrules
< \glxstrgeneralpuncrules
< \glxstrdigitrules
< \glxstrfractionrules
< \glxstrGeneralLatinIVrules
< \glxstrMathItalicGreekIrules
}
```

`xtrcontrolrules` These are control characters that are usually placed at the start of a rule in the ‘ignored characters’ section. These control characters are unlikely to appear in any entry fields but are provided for completeness. `\string` is used for punctuation characters in case they’ve been made active.

```
11822 \newcommand*\glxstrcontrolrules{%
```

```
11823 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
```

```
11824 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
```

```
11825 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
```

```
11826 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
```

```
11827 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
```

```
11828 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
```

```

11829 \string'\string=\glshex 0011
11830 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
11831 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
11832 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
11833 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
11834 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
11835 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
11836 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
11837 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
11838 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
11839 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
11840 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
11841 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
11842 }

```

**lsxtrspacerules** These are space characters.

```

11843 \newcommand*{\glsxtrspacerules}{%
11844 \string' \string'\string;
11845 \string'\glshex 00A0\string'\string;
11846 \string'\glshex 2000\string'\string;
11847 \string'\glshex 2001\string'\string;
11848 \string'\glshex 2002\string'\string;
11849 \string'\glshex 2003\string'\string;
11850 \string'\glshex 2004\string'\string;
11851 \string'\glshex 2005\string'\string;
11852 \string'\glshex 2006\string'\string;
11853 \string'\glshex 2007\string'\string;
11854 \string'\glshex 2008\string'\string;
11855 \string'\glshex 2009\string'\string;
11856 \string'\glshex 200A\string'\string;
11857 \string'\glshex 3000\string'
11858 }

```

**nonprintablerules** These are non-printable characters (BOM, tabs, line feed and carriage return).

```

11859 \newcommand*{\glsxtrnonprintablerules}{%
11860 \string'\glshex FEFF\string'\string;
11861 \string'\glshex 000A\string'\string;
11862 \string'\glshex 0009\string'\string;
11863 \string'\glshex 000C\string'\string;
11864 \string'\glshex 000B\string'
11865 }

```

**gdiacriticrules** Combining diacritic marks. This is split into multiple macros.

```

11866 \newcommand*{\glsxtrcombiningdiacriticrules}{%
11867 \glsxtrcombiningdiacriticIrules\string;
11868 \glsxtrcombiningdiacriticIIrules\string;
11869 \glsxtrcombiningdiacriticIIIrules\string;
11870 \glsxtrcombiningdiacriticIVrules
11871 }

```



diacriticIrules First set of combining diacritic marks.

```
11872 \newcommand*{\glxtrcombingdiacriticIrules}{%
11873 \glshex 0301\string;% combining acute
11874 \glshex 0300\string;% combining grave
11875 \glshex 0306\string;% combining breve
11876 \glshex 0302\string;% combining circumflex
11877 \glshex 030C\string;% combining caron
11878 \glshex 030A\string;% combining ring
11879 \glshex 030D\string;% combining vertical line above
11880 \glshex 0308\string;% combining diaeresis
11881 \glshex 030B\string;% combining double acute
11882 \glshex 0303\string;% combining tilde
11883 \glshex 0307\string;% combining dot above
11884 \glshex 0304% combining macron
11885 }
```

diacriticIIrules Second set of combining diacritic marks.

```
11886 \newcommand*{\glxtrcombingdiacriticIIrules}{%
11887 \glshex 0337\string;% combining short solidus overlay
11888 \glshex 0327\string;% combining cedilla
11889 \glshex 0328\string;% combining ogonek
11890 \glshex 0323\string;% combining dot below
11891 \glshex 0332\string;% combining low line
11892 \glshex 0305\string;% combining overline
11893 \glshex 0309\string;% combining hook above
11894 \glshex 030E\string;% combining double vertical line above
11895 \glshex 030F\string;% combining double grave accent
11896 \glshex 0310\string;% combining candrabindu
11897 \glshex 0311\string;% combining inverted breve
11898 \glshex 0312\string;% combining turned comma above
11899 \glshex 0313\string;% combining comma above
11900 \glshex 0314\string;% combining reversed comma above
11901 \glshex 0315\string;% combining comma above right
11902 \glshex 0316\string;% combining grave accent below
11903 \glshex 0317% combining acute accent below
11904 }
```

diacriticIIIrules Third set of combining diacritic marks.

```
11905 \newcommand*{\glxtrcombingdiacriticIIIrules}{%
11906 \glshex 0318\string;% combining left tack below
11907 \glshex 0319\string;% combining right tack below
11908 \glshex 031A\string;% combining left angle above
11909 \glshex 031B\string;% combining horn
11910 \glshex 031C\string;% combining left half ring below
11911 \glshex 031D\string;% combining up tack below
11912 \glshex 031E\string;% combining down tack below
11913 \glshex 031F\string;% combining plus sign below
11914 \glshex 0320\string;% combining minus sign below
11915 \glshex 0321\string;% combining palatalized hook below
```

11916 \glshex 0322\string;% combining retroflex hook below  
 11917 \glshex 0324\string;% combining diaresis below  
 11918 \glshex 0325\string;% combining ring below  
 11919 \glshex 0326\string;% combining comma below  
 11920 \glshex 0329\string;% combining vertical line below  
 11921 \glshex 032A\string;% combining bridge below  
 11922 \glshex 032B\string;% combining inverted double arch below  
 11923 \glshex 032C\string;% combining caron below  
 11924 \glshex 032D\string;% combining circumflex accent below  
 11925 \glshex 032E\string;% combining breve below  
 11926 \glshex 032F\string;% combining inverted breve below  
 11927 \glshex 0330\string;% combining tilde below  
 11928 \glshex 0331\string;% combining macron below  
 11929 \glshex 0333\string;% combining double low line  
 11930 \glshex 0334\string;% combining tilde overlay  
 11931 \glshex 0335\string;% combining short stroke overlay  
 11932 \glshex 0336\string;% combining long stroke overlay  
 11933 \glshex 0338\string;% combining long solidus overlay  
 11934 \glshex 0339\string;% combining combining right half ring below  
 11935 \glshex 033A\string;% combining inverted bridge below  
 11936 \glshex 033B\string;% combining square below  
 11937 \glshex 033C\string;% combining seagull below  
 11938 \glshex 033D\string;% combining x above  
 11939 \glshex 033E\string;% combining vertical tilde  
 11940 \glshex 033F\string;% combining double overline  
 11941 \glshex 0342\string;% combining Greek perispomeni  
 11942 \glshex 0344\string;% combining Greek dialytika tonos  
 11943 \glshex 0345\string;% combining Greek ypogegrammeni  
 11944 \glshex 0360\string;% combining double tilde  
 11945 \glshex 0361\string;% combining double inverted breve  
 11946 \glshex 0483\string;% combining Cyrillic titlo  
 11947 \glshex 0484\string;% combining Cyrillic palatalization  
 11948 \glshex 0485\string;% combining Cyrillic dasia pneumata  
 11949 \glshex 0486% combining Cyrillic psili pneumata  
 11950 }

iacriticIVrules Fourth set of combining diacritic marks.

11951 \newcommand\*{\glxtrcombiningdiacriticIVrules}{%  
 11952 \glshex 20D0\string;% combining left harpoon above  
 11953 \glshex 20D1\string;% combining right harpoon above  
 11954 \glshex 20D2\string;% combining long vertical line overlay  
 11955 \glshex 20D3\string;% combining short vertical line overlay  
 11956 \glshex 20D4\string;% combining anticlockwise arrow above  
 11957 \glshex 20D5\string;% combining clockwise arrow above  
 11958 \glshex 20D6\string;% combining left arrow above  
 11959 \glshex 20D7\string;% combining right arrow above  
 11960 \glshex 20D8\string;% combining ring overlay  
 11961 \glshex 20D9\string;% combining clockwise ring overlay  
 11962 \glshex 20DA\string;% combining anticlockwise ring overlay

```

11963 \glshex 20DB\string;% combining three dots above
11964 \glshex 20DC\string;% combining four dots above
11965 \glshex 20DD\string;% combining enclosing circle
11966 \glshex 20DE\string;% combining enclosing square
11967 \glshex 20DF\string;% combining enclosing diamond
11968 \glshex 20E0\string;% combining enclosing circle backslash
11969 \glshex 20E1% combining left right arrow above
11970 }

```

#### sxtrhyphenrules Hyphens.

```

11971 \newcommand*{\glsxtrhyphenrules}{%
11972 \string'\string-\string'\string;% ASCII hyphen
11973 \glshex 00AD\string;% soft hyphen
11974 \glshex 2010\string;% hyphen
11975 \glshex 2011\string;% non-breaking hyphen
11976 \glshex 2012\string;% figure dash
11977 \glshex 2013\string;% en dash
11978 \glshex 2014\string;% em dash
11979 \glshex 2015\string;% horizontal bar
11980 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
11981 }

```

#### eneralpuncrules General punctuation.

```

11982 \newcommand*{\glsxtrgeneralpuncrules}{%
11983 \glsxtrgeneralpuncIrules
11984 \string<\glsxtrcurrencyrules
11985 \string<\glsxtrgeneralpuncIIrules
11986 }

```

#### neralpuncIrules First set of general punctuation.

```

11987 \newcommand*{\glsxtrgeneralpuncIrules}{%
11988 \string'\glshex 005F\string'% underscore
11989 \string<\glshex 00AF% macron
11990 \string<\string'\glshex 002C\string'% comma
11991 \string<\string'\glshex 003B\string'% semi-colon
11992 \string<\string'\glshex 003A\string'% colon
11993 \string<\string'\glshex 0021\string'% exclamation mark
11994 \string<\glshex 00A1% inverted exclamation mark
11995 \string<\string'\glshex 003F\string'% question mark
11996 \string<\glshex 00BF% inverted question mark
11997 \string<\string'\glshex 002F\string'% solidus
11998 \string<\string'\glshex 002E\string'% full stop
11999 \string<\glshex 00B4% acute accent
12000 \string<\string'\glshex 0060\string'% grave accent
12001 \string<\string'\glshex 005E\string'% circumflex accent
12002 \string<\glshex 00A8% diaeresis
12003 \string<\string'\glshex 007E\string'% tilde
12004 \string<\glshex 00B7% middle dot
12005 \string<\glshex 00B8% cedilla

```

```

12006 \string<\string'\glshex 0027\string'% straight apostrophe
12007 \string<\string'\glshex 0022\string'% straight double quote
12008 \string<\glshex 00AB% left guillemet
12009 \string<\glshex 00BB% right guillemet
12010 \string<\string'\glshex 0028\string'% left parenthesis
12011 \string=<\glshex 207D\string=<\glshex 208D% super/subscript left parenthesis
12012 \string<\string'\glshex 0029\string'% right parenthesis
12013 \string=<\glshex 207E\string=<\glshex 208E% super/subscript right parenthesis
12014 \string<\string'\glshex 005B\string'% left square bracket
12015 \string<\string'\glshex 005D\string'% right square bracket
12016 \string<\string'\glshex 007B\string'% left curly bracket
12017 \string<\string'\glshex 007D\string'% right curly bracket
12018 \string<\glshex 00A7% section sign
12019 \string<\glshex 00B6% pilcrow sign
12020 \string<\glshex 00A9% copyright sign
12021 \string<\glshex 00AE% registered sign
12022 \string<\string'\glshex 0040\string'% at sign
12023 }

```

trcurrencyrules General punctuation.

```

12024 \newcommand*{\glxtrcurrencyrules}{%
12025 \glshex 00A4% currency sign
12026 \string<\glshex 0E3F% Thai currency symbol baht
12027 \string<\glshex 00A2% cent sign
12028 \string<\glshex 20A1% colon sign
12029 \string<\glshex 20A2% cruzeiro sign
12030 \string<\string'\glshex 0024\string'% dollar sign
12031 \string<\glshex 20AB% dong sign
12032 \string<\glshex 20AC% euro sign
12033 \string<\glshex 20A3% French franc sign
12034 \string<\glshex 20A4% lira sign
12035 \string<\glshex 20A5% mill sign
12036 \string<\glshex 20A6% naira sign
12037 \string<\glshex 20A7% peseta sign
12038 \string<\glshex 00A3% pound sign
12039 \string<\glshex 20A8% rupee sign
12040 \string<\glshex 20AA% new sheqel sign
12041 \string<\glshex 20A9% won sign
12042 \string<\glshex 00A5% yen sign
12043 }

```

eralpuncIIrules Second set of general punctuation.

```

12044 \newcommand*{\glxtrgeneralpuncIIrules}{%
12045 \string'\glshex 002A\string'% asterisk
12046 \string<\string'\glshex 005C\string'% backslash
12047 \string<\string'\glshex 0026\string'% ampersand
12048 \string<\string'\glshex 0023\string'% hash sign
12049 \string<\string'\glshex 0025\string'% percent sign
12050 \string<\string'\glshex 002B\string'% plus sign

```

```

12051 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
12052 \string<\glshex 00B1% plus-minus sign
12053 \string<\glshex 00F7% division sign
12054 \string<\glshex 00D7% multiplication sign
12055 \string<\string'\glshex 003C\string'% less-than sign
12056 \string<\string'\glshex 003D\string'% equals sign
12057 \string<\string'\glshex 003E\string'% greater-than sign
12058 \string<\glshex 00AC% not sign
12059 \string<\string'\glshex 007C\string'% vertical bar (pipe)
12060 \string<\glshex 00A6% broken bar
12061 \string<\glshex 00B0% degree sign
12062 \string<\glshex 00B5% micron sign
12063 }

```

eralLatinIrules Basic Latin alphabet.

```

12064 \newcommand*{\glxtrGeneralLatinIrules}{%
12065 \glxtrLatinA
12066 \string<b,B%
12067 \string<c,C%
12068 \string<d,D%
12069 \string<\glxtrLatinE
12070 \string<f,F%
12071 \string<g,G%
12072 \string<\glxtrLatinH
12073 \string<\glxtrLatinI
12074 \string<j,J%
12075 \string<\glxtrLatinK
12076 \string<\glxtrLatinL
12077 \string<\glxtrLatinM
12078 \string<\glxtrLatinN
12079 \string<\glxtrLatinO
12080 \string<\glxtrLatinP
12081 \string<q,Q%
12082 \string<r,R%
12083 \string<\glxtrLatinS
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 }

```

ralLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

12092 \newcommand*{\glxtrGeneralLatinIIrules}{%
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& SS \string, \glxtrLatinEszettSs
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 }

```

**allLatinIIIrules** General Latin alphabet (eth between D and E, ß treated as SZ).

```

12122 \newcommand*{\glxtrGeneralLatinIIIrules}{%
12123 \glxtrLatinA
12124 \string<b,B%
12125 \string<c,C%
12126 \string<d,D%
12127 \string<\glxtrLatinEth
12128 \string<\glxtrLatinE
12129 \string<f,F%
12130 \string<g,G%
12131 \string<\glxtrLatinH
12132 \string<\glxtrLatinI
12133 \string<j,J%
12134 \string<\glxtrLatinK
12135 \string<\glxtrLatinL
12136 \string<\glxtrLatinM
12137 \string<\glxtrLatinN
12138 \string<\glxtrLatinO
12139 \string<\glxtrLatinP
12140 \string<q,Q%
12141 \string<r,R%
12142 \string<\glxtrLatinS

```

```

12143 \string& SZ, \glxtrLatinEszettSz
12144 \string<\glxtrLatinT
12145 \string<u,U%
12146 \string<v,V%
12147 \string<w,W%
12148 \string<\glxtrLatinX
12149 \string<y,Y%
12150 \string<z,Z%
12151 }

```

**GeneralLatinIVrules** General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```

12152 \newcommand*{\glxtrGeneralLatinIVrules}{%
12153 \glxtrLatinA
12154 \string& AE , \glxtrLatinAELigature
12155 \string<b,B%
12156 \string<c,C%
12157 \string<d,D%
12158 \string<\glxtrLatinEth
12159 \string<\glxtrLatinE
12160 \string<f,F%
12161 \string<g,G%
12162 \string<\glxtrLatinH
12163 \string<\glxtrLatinI
12164 \string<j,J%
12165 \string<\glxtrLatinK
12166 \string<\glxtrLatinL
12167 \string<\glxtrLatinM
12168 \string<\glxtrLatinN
12169 \string<\glxtrLatinO
12170 \string& OE , \glxtrLatinOELigature
12171 \string<\glxtrLatinP
12172 \string<q,Q%
12173 \string<r,R%
12174 \string<\glxtrLatinS
12175 \string& SS , \glxtrLatinEszettSs
12176 \string<\glxtrLatinT
12177 \string& th =\glshex 00DE
12178 \string& TH =\glshex 00FE
12179 \string<u,U%
12180 \string<v,V%
12181 \string<w,W%
12182 \string<\glxtrLatinX
12183 \string<y,Y%
12184 \string<z,Z%
12185 }

```

**GeneralLatinVrules** General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

12186 \newcommand*{\glxtrGeneralLatinVrules}{%

```

```

12187 \glxtrLatinA
12188 \string<b,B%
12189 \string<c,C%
12190 \string<d,D%
12191 \string<\glxtrLatinEth
12192 \string<\glxtrLatinE
12193 \string<f,F%
12194 \string<g,G%
12195 \string<\glxtrLatinH
12196 \string<\glxtrLatinI
12197 \string<j,J%
12198 \string<\glxtrLatinK
12199 \string<\glxtrLatinL
12200 \string<\glxtrLatinM
12201 \string<\glxtrLatinN
12202 \string<\glxtrLatinO
12203 \string<\glxtrLatinP
12204 \string<q,Q%
12205 \string<r,R%
12206 \string<\glxtrLatinS
12207 \string& SS , \glxtrLatinEszettSs
12208 \string<\glxtrLatinT
12209 \string& th =\glshex 00DE
12210 \string& TH =\glshex 00FE
12211 \string<u,U%
12212 \string<v,V%
12213 \string<w,W%
12214 \string<\glxtrLatinX
12215 \string<y,Y%
12216 \string<z,Z%
12217 }

```

**raLatinVIrules** General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

12218 \newcommand*{\glxtrGeneralLatinVIrules}{%
12219 \glxtrLatinA
12220 \string<b,B%
12221 \string<c,C%
12222 \string<d,D%
12223 \string<\glxtrLatinEth
12224 \string<\glxtrLatinE
12225 \string<f,F%
12226 \string<g,G%
12227 \string<\glxtrLatinH
12228 \string<\glxtrLatinI
12229 \string<j,J%
12230 \string<\glxtrLatinK
12231 \string<\glxtrLatinL
12232 \string<\glxtrLatinM
12233 \string<\glxtrLatinN

```



```

12234 \string<\glxtrLatinO
12235 \string<\glxtrLatinP
12236 \string<q,Q%
12237 \string<r,R%
12238 \string<\glxtrLatinS
12239 \string& SZ , \glxtrLatinEszettSz
12240 \string<\glxtrLatinT
12241 \string& th =\glshex 00DE
12242 \string& TH =\glshex 00FE
12243 \string<u,U%
12244 \string<v,V%
12245 \string<w,W%
12246 \string<\glxtrLatinX
12247 \string<y,Y%
12248 \string<z,Z%
12249 }

```

allLatinVIIrules General Latin alphabet (Æ between A and B, eth between D and E, insular G as G, CE between O and P, long S equivalent to S, Þ between T and U and wynn as W).

```

12250 \newcommand*{\glxtrGeneralLatinVIIrules}{%
12251 \glxtrLatinA
12252 \string<\glxtrLatinAELigature
12253 \string<b,B%
12254 \string<c,C%
12255 \string<d,D%
12256 \string<\glxtrLatinEth
12257 \string<\glxtrLatinE
12258 \string<f,F%
12259 \string<\glxtrLatinInsularG
12260 \string<\glxtrLatinH
12261 \string<\glxtrLatinI
12262 \string<j,J%
12263 \string<\glxtrLatinK
12264 \string<\glxtrLatinL
12265 \string<\glxtrLatinM
12266 \string<\glxtrLatinN
12267 \string<\glxtrLatinO
12268 \string<\glxtrLatinOELigature
12269 \string<\glxtrLatinP
12270 \string<q,Q%
12271 \string<r,R%
12272 \string<\glshex 017F=\glxtrLatinS % s and long s
12273 \string<\glxtrLatinT
12274 \string<\glxtrLatinThorn
12275 \string<u,U%
12276 \string<v,V%
12277 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
12278 \string<\glxtrLatinX
12279 \string<y,Y%

```

```

12280 \string<z,Z%
12281 }

```

`\LatinVIIIrules` General Latin alphabet ( $\mathring{A}$  treated as AE and  $\mathring{C}$  treated as OE,  $\mathring{P}$  treated as TH,  $\mathring{B}$  treated as SS,  $\mathring{eth}$  treated as D,  $\mathring{O}$  treated as O,  $\mathring{L}$  treated as L).

```

12282 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
12283 \glxtrLatinA
12284 \string& AE , \glxtrLatinAELigature
12285 \string<b,B%
12286 \string<c,C%
12287 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
12288 \string<\glxtrLatinE
12289 \string<f,F%
12290 \string<g,G%
12291 \string<\glxtrLatinH
12292 \string<\glxtrLatinI
12293 \string<j,J%
12294 \string<\glxtrLatinK
12295 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
12296 \string<\glxtrLatinM
12297 \string<\glxtrLatinN
12298 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
12299 \string& OE , \glxtrLatinOELigature
12300 \string<\glxtrLatinP
12301 \string<q,Q%
12302 \string<r,R%
12303 \string<\glxtrLatinS
12304 \string& SS , \glxtrLatinEszettSs
12305 \string<\glxtrLatinT
12306 \string& th =\glshex 00DE
12307 \string& TH =\glshex 00FE
12308 \string<u,U%
12309 \string<v,V%
12310 \string<w,W%
12311 \string<\glxtrLatinX
12312 \string<y,Y%
12313 \string<z,Z%
12314 }

```

`\glxtrLatinA`

```

12315 \newcommand*{\glxtrLatinA}{%
12316 a\string=\glshex 00AA\string=\glshex 2090,A
12317 }

```

`\glxtrLatinE`

```

12318 \newcommand*{\glxtrLatinE}{%
12319 e\string=\glshex 2091,E
12320 }

```

\glsxtrLatinH

```
12321 \newcommand*{\glsxtrLatinH}{%
12322   h\string=\glshex 2095,H
12323 }
```

\glsxtrLatinI

```
12324 \newcommand*{\glsxtrLatinI}{%
12325   i\string=\glshex 2071,I
12326 }
```

\glsxtrLatinK

```
12327 \newcommand*{\glsxtrLatinK}{%
12328   k\string=\glshex 2096,K
12329 }
```

\glsxtrLatinL

```
12330 \newcommand*{\glsxtrLatinL}{%
12331   l\string=\glshex 2097,L
12332 }
```

\glsxtrLatinM

```
12333 \newcommand*{\glsxtrLatinM}{%
12334   m\string=\glshex 2098,M
12335 }
```

\glsxtrLatinN

```
12336 \newcommand*{\glsxtrLatinN}{%
12337   n\string=\glshex 207F\string=\glshex 2099,N
12338 }
```

\glsxtrLatinO

```
12339 \newcommand*{\glsxtrLatinO}{%
12340   o\string=\glshex 00BA\string=\glshex 2092,O
12341 }
```

\glsxtrLatinP

```
12342 \newcommand*{\glsxtrLatinP}{%
12343   p\string=\glshex 209A,P
12344 }
```

\glsxtrLatinS

```
12345 \newcommand*{\glsxtrLatinS}{%
12346   s\string=\glshex 209B,S
12347 }
```

\glsxtrLatinT

```
12348 \newcommand*{\glsxtrLatinT}{%
12349   t\string=\glshex 209C,T
12350 }
```

```

\glxtrLatinX
12351 \newcommand*{\glxtrLatinX}{%
12352   x\string=\glshex 2093,X
12353 }

lsxtrLatinSchwa  Latin schwa (lower case, subscript and upper case).
12354 \newcommand*{\glxtrLatinSchwa}{%
12355   \glshex 0259\string=\glshex 2094,\glshex 018F
12356 }

trLatinEszettSs
12357 \newcommand*{\glxtrLatinEszettSs}{%
12358   \glshex 00DF% eszett
12359   \string=\glshex 017Fs % long S s
12360 }

trLatinEszettSz
12361 \newcommand*{\glxtrLatinEszettSz}{%
12362   \glshex 00DF% eszett
12363   \string= \glshex 017Fz % long S z
12364 }

\glxtrLatinEth
12365 \newcommand*{\glxtrLatinEth}{%
12366   \glshex 00F0,\glshex 00D0% eth
12367 }

lsxtrLatinThorn
12368 \newcommand*{\glxtrLatinThorn}{%
12369   \glshex 00FE,\glshex 00DE% thorn
12370 }

LatinAELigature
12371 \newcommand*{\glxtrLatinAELigature}{%
12372   \glshex 00E6,\glshex 00C6% AE-ligature
12373 }

LatinOELigature
12374 \newcommand*{\glxtrLatinOELigature}{%
12375   \glshex 0153,\glshex 0152% OE-ligature
12376 }

\glxtrLatinAA
12377 \newcommand*{\glxtrLatinAA}{%
12378   \glshex 00E5=a\glshex 030A,% \aa
12379   \glshex 00C5=A\glshex 030A% \AA
12380 }

```

glsxtrLatinWynn

```
12381 \newcommand*{\glsxtrLatinWynn}{%  
12382   \glsheX 01BF,\glsheX 01F7% wynn  
12383 }
```

trLatinInsularG

```
12384 \newcommand*{\glsxtrLatinInsularG}{%  
12385   \glsheX 1D79,\glsheX A77D% insular G  
12386   \string; g, G  
12387 }
```

sxtrLatinOslash

```
12388 \newcommand*{\glsxtrLatinOslash}{%  
12389   \glsheX 00F8,\glsheX 00D8% \o, \O  
12390 }
```

sxtrLatinLslash

```
12391 \newcommand*{\glsxtrLatinLslash}{%  
12392   \glsheX 0142,\glsheX 0141% \l, \L  
12393 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
12394 \newcommand*{\glsxtrMathUpGreekIrules}{%  
12395   \glsxtrUpAlpha  
12396   \string<\glsxtrUpBeta  
12397   \string<\glsxtrUpGamma  
12398   \string<\glsxtrUpDelta  
12399   \string<\glsxtrUpEpsilon  
12400   \string<\glsxtrUpDigamma  
12401   \string<\glsxtrUpZeta  
12402   \string<\glsxtrUpEta  
12403   \string<\glsxtrUpTheta  
12404   \string<\glsxtrUpIota  
12405   \string<\glsxtrUpKappa  
12406   \string<\glsxtrUpLambda  
12407   \string<\glsxtrUpMu  
12408   \string<\glsxtrUpNu  
12409   \string<\glsxtrUpXi  
12410   \string<\glsxtrUpOmicron  
12411   \string<\glsxtrUpPi  
12412   \string<\glsxtrUpRho  
12413   \string<\glsxtrUpSigma  
12414   \string<\glsxtrUpTau  
12415   \string<\glsxtrUpUpsilon  
12416   \string<\glsxtrUpPhi  
12417   \string<\glsxtrUpChi  
12418   \string<\glsxtrUpPsi  
12419   \string<\glsxtrUpOmega  
12420 }
```

hUpGreekIIrules Doesn't include digamma.

```
12421 \newcommand*{\glxtrMathUpGreekIIrules}{%
12422   \glxtrUpAlpha
12423   \string<\glxtrUpBeta
12424   \string<\glxtrUpGamma
12425   \string<\glxtrUpDelta
12426   \string<\glxtrUpEpsilon
12427   \string<\glxtrUpZeta
12428   \string<\glxtrUpEta
12429   \string<\glxtrUpTheta
12430   \string<\glxtrUpIota
12431   \string<\glxtrUpKappa
12432   \string<\glxtrUpLambda
12433   \string<\glxtrUpMu
12434   \string<\glxtrUpNu
12435   \string<\glxtrUpXi
12436   \string<\glxtrUpOmicron
12437   \string<\glxtrUpPi
12438   \string<\glxtrUpRho
12439   \string<\glxtrUpSigma
12440   \string<\glxtrUpTau
12441   \string<\glxtrUpUpsilon
12442   \string<\glxtrUpPhi
12443   \string<\glxtrUpChi
12444   \string<\glxtrUpPsi
12445   \string<\glxtrUpOmega
12446 }
```

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.

```
12447 \newcommand*{\glxtrMathItalicGreekIrules}{%
12448   \glxtrMathItalicAlpha
12449   \string<\glxtrMathItalicBeta
12450   \string<\glxtrMathItalicGamma
12451   \string<\glxtrMathItalicDelta
12452   \string<\glxtrMathItalicEpsilon
12453   \string<\glxtrUpDigamma
12454   \string<\glxtrMathItalicZeta
12455   \string<\glxtrMathItalicEta
12456   \string<\glxtrMathItalicTheta
12457   \string<\glxtrMathItalicIota
12458   \string<\glxtrMathItalicKappa
12459   \string<\glxtrMathItalicLambda
12460   \string<\glxtrMathItalicMu
12461   \string<\glxtrMathItalicNu
12462   \string<\glxtrMathItalicXi
12463   \string<\glxtrMathItalicOmicron
12464   \string<\glxtrMathItalicPi
12465   \string<\glxtrMathItalicRho
```

```

12466 \string<\glxtrMathItalicSigma
12467 \string<\glxtrMathItalicTau
12468 \string<\glxtrMathItalicUpsilon
12469 \string<\glxtrMathItalicPhi
12470 \string<\glxtrMathItalicChi
12471 \string<\glxtrMathItalicPsi
12472 \string<\glxtrMathItalicOmega
12473 }

```

**LowerGreekIIrules** Doesn't include digamma.

```

12474 \newcommand*{\glxtrMathItalicGreekIIrules}{%
12475 \glxtrMathItalicAlpha
12476 \string<\glxtrMathItalicBeta
12477 \string<\glxtrMathItalicGamma
12478 \string<\glxtrMathItalicDelta
12479 \string<\glxtrMathItalicEpsilon
12480 \string<\glxtrMathItalicZeta
12481 \string<\glxtrMathItalicEta
12482 \string<\glxtrMathItalicTheta
12483 \string<\glxtrMathItalicIota
12484 \string<\glxtrMathItalicKappa
12485 \string<\glxtrMathItalicLambda
12486 \string<\glxtrMathItalicMu
12487 \string<\glxtrMathItalicNu
12488 \string<\glxtrMathItalicXi
12489 \string<\glxtrMathItalicOmicron
12490 \string<\glxtrMathItalicPi
12491 \string<\glxtrMathItalicRho
12492 \string<\glxtrMathItalicSigma
12493 \string<\glxtrMathItalicTau
12494 \string<\glxtrMathItalicUpsilon
12495 \string<\glxtrMathItalicPhi
12496 \string<\glxtrMathItalicChi
12497 \string<\glxtrMathItalicPsi
12498 \string<\glxtrMathItalicOmega
12499 }

```

**UpperGreekIrules** Upper case only (includes upright digamma).

```

12500 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
12501 \glshex 1D6E2% upper case alpha (maths italic)
12502 \string<\glshex 1D6E3% upper case beta (maths italic)
12503 \string<\glshex 1D6E4% upper case gamma (maths italic)
12504 \string<\glshex 1D6E5% upper case delta (maths italic)
12505 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12506 \string<\glshex 03DC% upper case digamma
12507 \string<\glshex 1D6E7% upper case zeta (maths italic)
12508 \string<\glshex 1D6E8% upper case eta (maths italic)
12509 \string<\glshex 1D6E9% upper case theta (maths italic)
12510 \string=<\glshex 1D6F3% upper case theta variant (maths italic)

```

```

12511 \string<\glshex 1D6EA% upper case iota (maths italic)
12512 \string<\glshex 1D6EB% upper case kappa (maths italic)
12513 \string<\glshex 1D6EC% upper case lambda (maths italic)
12514 \string<\glshex 1D6ED% upper case mu (maths italic)
12515 \string<\glshex 1D6EE% upper case nu (maths italic)
12516 \string<\glshex 1D6EF% upper case xi (maths italic)
12517 \string<\glshex 1D6F0% upper case omicron (maths italic)
12518 \string<\glshex 1D6F1% upper case pi (maths italic)
12519 \string<\glshex 1D6F2% upper case rho (maths italic)
12520 \string<\glshex 1D6F4% upper case sigma (maths italic)
12521 \string<\glshex 1D6F5% upper case tau (maths italic)
12522 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12523 \string<\glshex 1D6F7% upper case phi (maths italic)
12524 \string<\glshex 1D6F8% upper case chi (maths italic)
12525 \string<\glshex 1D6F9% upper case psi (maths italic)
12526 \string<\glshex 1D6FA% upper case omega (maths italic)
12527 }

```

**perGreekIIrules** Upper case only (doesn't include upright digamma).

```

12528 \newcommand*{\glsxtrMathItalicUpperGreekIIrules}{%
12529 \glshex 1D6E2% upper case alpha (maths italic)
12530 \string<\glshex 1D6E3% upper case beta (maths italic)
12531 \string<\glshex 1D6E4% upper case gamma (maths italic)
12532 \string<\glshex 1D6E5% upper case delta (maths italic)
12533 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12534 \string<\glshex 1D6E7% upper case zeta (maths italic)
12535 \string<\glshex 1D6E8% upper case eta (maths italic)
12536 \string<\glshex 1D6E9% upper case theta (maths italic)
12537 \string<\glshex 1D6F3% upper case theta variant (maths italic)
12538 \string<\glshex 1D6EA% upper case iota (maths italic)
12539 \string<\glshex 1D6EB% upper case kappa (maths italic)
12540 \string<\glshex 1D6EC% upper case lambda (maths italic)
12541 \string<\glshex 1D6ED% upper case mu (maths italic)
12542 \string<\glshex 1D6EE% upper case nu (maths italic)
12543 \string<\glshex 1D6EF% upper case xi (maths italic)
12544 \string<\glshex 1D6F0% upper case omicron (maths italic)
12545 \string<\glshex 1D6F1% upper case pi (maths italic)
12546 \string<\glshex 1D6F2% upper case rho (maths italic)
12547 \string<\glshex 1D6F4% upper case sigma (maths italic)
12548 \string<\glshex 1D6F5% upper case tau (maths italic)
12549 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12550 \string<\glshex 1D6F7% upper case phi (maths italic)
12551 \string<\glshex 1D6F8% upper case chi (maths italic)
12552 \string<\glshex 1D6F9% upper case psi (maths italic)
12553 \string<\glshex 1D6FA% upper case omega (maths italic)
12554 }

```

**lowerGreekIrules** Lower case only (includes upright digamma).

```

12555 \newcommand*{\glsxtrMathItalicLowerGreekIrules}{%

```



```

12556 \glshex 1D6FC% lower case alpha (maths italic)
12557 \string<\glshex 1D6FD% lower case beta (maths italic)
12558 \string<\glshex 1D6FE% lower case gamma (maths italic)
12559 \string<\glshex 1D6FF% lower case delta (maths italic)
12560 \string<\glshex 1D700% lower case epsilon (maths italic)
12561 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12562 \string<\glshex 03DD% lower case digamma
12563 \string<\glshex 1D701% lower case zeta (maths italic)
12564 \string<\glshex 1D702% lower case eta (maths italic)
12565 \string<\glshex 1D703% lower case theta (maths italic)
12566 \string=\glshex 1D717% lower case theta variant (maths italic)
12567 \string<\glshex 1D704% lower case iota (maths italic)
12568 \string<\glshex 1D705% lower case kappa (maths italic)
12569 \string=\glshex 1D718% lower case kappa variant (maths italic)
12570 \string<\glshex 1D706% lower case lambda (maths italic)
12571 \string<\glshex 1D707% lower case mu (maths italic)
12572 \string<\glshex 1D708% lower case nu (maths italic)
12573 \string<\glshex 1D709% lower case xi (maths italic)
12574 \string<\glshex 1D70A% lower case omicron (maths italic)
12575 \string<\glshex 1D70B% lower case pi (maths italic)
12576 \string=\glshex 1D71B% lower case pi variant (maths italic)
12577 \string<\glshex 1D70C% lower case rho (maths italic)
12578 \string=\glshex 1D71A% lower case rho variant (maths italic)
12579 \string<\glshex 1D70D% lower case final sigma (maths italic)
12580 \string=\glshex 1D70E% lower case sigma (maths italic)
12581 \string<\glshex 1D70F% lower case tau (maths italic)
12582 \string<\glshex 1D710% lower case upsilon (maths italic)
12583 \string<\glshex 1D711% lower case phi (maths italic)
12584 \string=\glshex 1D719% lower case phi variant (maths italic)
12585 \string<\glshex 1D712% lower case chi (maths italic)
12586 \string<\glshex 1D713% lower case psi (maths italic)
12587 \string<\glshex 1D714% lower case omega (maths italic)
12588 }

```

LowerGreekIIrules Lower case only (doesn't includes upright digamma).

```

12589 \newcommand*{\glxtrMathItalicLowerGreekIIrules}{%
12590 \glshex 1D6FC% lower case alpha (maths italic)
12591 \string<\glshex 1D6FD% lower case beta (maths italic)
12592 \string<\glshex 1D6FE% lower case gamma (maths italic)
12593 \string<\glshex 1D6FF% lower case delta (maths italic)
12594 \string<\glshex 1D700% lower case epsilon (maths italic)
12595 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12596 \string<\glshex 1D701% lower case zeta (maths italic)
12597 \string<\glshex 1D702% lower case eta (maths italic)
12598 \string<\glshex 1D703% lower case theta (maths italic)
12599 \string=\glshex 1D717% lower case theta variant (maths italic)
12600 \string<\glshex 1D704% lower case iota (maths italic)
12601 \string<\glshex 1D705% lower case kappa (maths italic)
12602 \string=\glshex 1D718% lower case kappa variant (maths italic)

```

```

12603 \string<\glshex 1D706% lower case lambda (maths italic)
12604 \string<\glshex 1D707% lower case mu (maths italic)
12605 \string<\glshex 1D708% lower case nu (maths italic)
12606 \string<\glshex 1D709% lower case xi (maths italic)
12607 \string<\glshex 1D70A% lower case omicron (maths italic)
12608 \string<\glshex 1D70B% lower case pi (maths italic)
12609 \string=\glshex 1D71B% lower case pi variant (maths italic)
12610 \string<\glshex 1D70C% lower case rho (maths italic)
12611 \string=\glshex 1D71A% lower case rho variant (maths italic)
12612 \string<\glshex 1D70D% lower case final sigma (maths italic)
12613 \string=\glshex 1D70E% lower case sigma (maths italic)
12614 \string<\glshex 1D70F% lower case tau (maths italic)
12615 \string<\glshex 1D710% lower case upsilon (maths italic)
12616 \string<\glshex 1D711% lower case phi (maths italic)
12617 \string=\glshex 1D719% lower case phi variant (maths italic)
12618 \string<\glshex 1D712% lower case chi (maths italic)
12619 \string<\glshex 1D713% lower case psi (maths italic)
12620 \string<\glshex 1D714% lower case omega (maths italic)
12621 }

```

**MathGreekIrules** Includes both upright and italic with digamma between epsilon and zeta.

```

12622 \newcommand*{\glxtrMathGreekIrules}{%
12623 \glxtrMathItalicAlpha
12624 \string;\glxtrUpAlpha
12625 \string<\glxtrMathItalicBeta
12626 \string;\glxtrUpBeta
12627 \string<\glxtrMathItalicGamma
12628 \string;\glxtrUpGamma
12629 \string<\glxtrMathItalicDelta
12630 \string;\glxtrUpDelta
12631 \string<\glxtrMathItalicEpsilon
12632 \string;\glxtrUpEpsilon
12633 \string<\glxtrUpDigamma
12634 \string<\glxtrMathItalicZeta
12635 \string;\glxtrUpZeta
12636 \string<\glxtrMathItalicEta
12637 \string;\glxtrUpEta
12638 \string<\glxtrMathItalicTheta
12639 \string;\glxtrUpTheta
12640 \string<\glxtrMathItalicIota
12641 \string;\glxtrUpIota
12642 \string<\glxtrMathItalicKappa
12643 \string;\glxtrUpKappa
12644 \string<\glxtrMathItalicLambda
12645 \string;\glxtrUpLambda
12646 \string<\glxtrMathItalicMu
12647 \string;\glxtrUpMu
12648 \string<\glxtrMathItalicNu
12649 \string;\glxtrUpNu

```

```

12650 \string<\glxtrMathItalicXi
12651 \string;\glxtrUpXi
12652 \string<\glxtrMathItalicOmicron
12653 \string;\glxtrUpOmicron
12654 \string<\glxtrMathItalicPi
12655 \string;\glxtrUpPi
12656 \string<\glxtrMathItalicRho
12657 \string;\glxtrUpRho
12658 \string<\glxtrMathItalicSigma
12659 \string;\glxtrUpSigma
12660 \string<\glxtrMathItalicTau
12661 \string;\glxtrUpTau
12662 \string<\glxtrMathItalicUpsilon
12663 \string;\glxtrUpUpsilon
12664 \string<\glxtrMathItalicPhi
12665 \string;\glxtrUpPhi
12666 \string<\glxtrMathItalicChi
12667 \string;\glxtrUpChi
12668 \string<\glxtrMathItalicPsi
12669 \string;\glxtrUpPsi
12670 \string<\glxtrMathItalicOmega
12671 \string;\glxtrUpOmega
12672 }

```

**mathGreekIIrules** Includes both upright and italic (digamma not included).

```

12673 \newcommand*{\glxtrMathGreekIIrules}{%
12674 \glxtrMathItalicAlpha
12675 \string;\glxtrUpAlpha
12676 \string<\glxtrMathItalicBeta
12677 \string;\glxtrUpBeta
12678 \string<\glxtrMathItalicGamma
12679 \string;\glxtrUpGamma
12680 \string<\glxtrMathItalicDelta
12681 \string;\glxtrUpDelta
12682 \string<\glxtrMathItalicEpsilon
12683 \string;\glxtrUpEpsilon
12684 \string<\glxtrMathItalicZeta
12685 \string;\glxtrUpZeta
12686 \string<\glxtrMathItalicEta
12687 \string;\glxtrUpEta
12688 \string<\glxtrMathItalicTheta
12689 \string;\glxtrUpTheta
12690 \string<\glxtrMathItalicIota
12691 \string;\glxtrUpIota
12692 \string<\glxtrMathItalicKappa
12693 \string;\glxtrUpKappa
12694 \string<\glxtrMathItalicLambda
12695 \string;\glxtrUpLambda
12696 \string<\glxtrMathItalicMu

```

```

12697 \string;\glxtrUpMu
12698 \string<\glxtrMathItalicNu
12699 \string;\glxtrUpNu
12700 \string<\glxtrMathItalicXi
12701 \string;\glxtrUpXi
12702 \string<\glxtrMathItalicOmicron
12703 \string;\glxtrUpOmicron
12704 \string<\glxtrMathItalicPi
12705 \string;\glxtrUpPi
12706 \string<\glxtrMathItalicRho
12707 \string;\glxtrUpRho
12708 \string<\glxtrMathItalicSigma
12709 \string;\glxtrUpSigma
12710 \string<\glxtrMathItalicTau
12711 \string;\glxtrUpTau
12712 \string<\glxtrMathItalicUpsilon
12713 \string;\glxtrUpUpsilon
12714 \string<\glxtrMathItalicPhi
12715 \string;\glxtrUpPhi
12716 \string<\glxtrMathItalicChi
12717 \string;\glxtrUpChi
12718 \string<\glxtrMathItalicPsi
12719 \string;\glxtrUpPsi
12720 \string<\glxtrMathItalicOmega
12721 \string;\glxtrUpOmega
12722 }

```

`\glxtrUpAlpha`

```

12723 \newcommand*{\glxtrUpAlpha}{%
12724   \glshex 03B1,% lower case alpha
12725   \glshex 0391% upper case alpha
12726 }

```

`\glxtrUpBeta`

```

12727 \newcommand*{\glxtrUpBeta}{%
12728   \glshex 03B2,% lower case beta
12729   \glshex 0392% upper case beta
12730 }

```

`\glxtrUpGamma`

```

12731 \newcommand*{\glxtrUpGamma}{%
12732   \glshex 03B3,% lower case gamma
12733   \glshex 0393% upper case gamma
12734 }

```

`\glxtrUpDelta`

```

12735 \newcommand*{\glxtrUpDelta}{%
12736   \glshex 03B4,% lower case delta
12737   \glshex 0394% upper case delta

```

12738 }

glsxtrUpEpsilon

```
12739 \newcommand*{\glsxtrUpEpsilon}{%
12740   \glsheX 03B5% lower case epsilon
12741   \string=\glsheX 03F5,% lower case epsilon variant
12742   \glsheX 0395% upper case epsilon
12743 }
```

glsxtrUpDigamma

```
12744 \newcommand*{\glsxtrUpDigamma}{%
12745   \glsheX 03DD,% lower case digamma
12746   \glsheX 03DC% upper case digamma
12747 }
```

\glsxtrUpZeta

```
12748 \newcommand*{\glsxtrUpZeta}{%
12749   \glsheX 03B6,% lower case zeta
12750   \glsheX 0396% upper case zeta
12751 }
```

\glsxtrUpEta

```
12752 \newcommand*{\glsxtrUpEta}{%
12753   \glsheX 03B7,% lower case eta
12754   \glsheX 0397% upper case eta
12755 }
```

\glsxtrUpTheta

```
12756 \newcommand*{\glsxtrUpTheta}{%
12757   \glsheX 03B8% lower case theta
12758   \string=\glsheX 03D1,% lower case theta variant
12759   \glsheX 0398% upper case theta
12760 }
```

\glsxtrUpIota

```
12761 \newcommand*{\glsxtrUpIota}{%
12762   \glsheX 03B9,% lower case iota
12763   \glsheX 0399% upper case iota
12764 }
```

\glsxtrUpKappa

```
12765 \newcommand*{\glsxtrUpKappa}{%
12766   \glsheX 03BA% lower case kappa
12767   \string=\glsheX 03F0,% lower case kappa variant
12768   \glsheX 039A% upper case kappa
12769 }
```

\glxtrUpLambda

```
12770 \newcommand*{\glxtrUpLambda}{%
12771   \glshex 03BB,% lower lambda
12772   \glshex 039B% upper case lambda
12773 }
```

\glxtrUpMu

```
12774 \newcommand*{\glxtrUpMu}{%
12775   \glshex 03BC,% lower case mu
12776   \glshex 039C% upper case mu
12777 }
```

\glxtrUpNu

```
12778 \newcommand*{\glxtrUpNu}{%
12779   \glshex 03BD,% lower case nu
12780   \glshex 039D% upper case nu
12781 }
```

\glxtrUpXi

```
12782 \newcommand*{\glxtrUpXi}{%
12783   \glshex 03BE,% lower case xi
12784   \glshex 039E% upper case xi
12785 }
```

\glxtrUpOmicron

```
12786 \newcommand*{\glxtrUpOmicron}{%
12787   \glshex 03BF,% lower case omicron
12788   \glshex 039F% upper case omicron
12789 }
```

\glxtrUpPi

```
12790 \newcommand*{\glxtrUpPi}{%
12791   \glshex 03C0% lower case pi
12792   \string=\glshex 03D6,% lower case pi variant
12793   \glshex 03A0% upper case pi
12794 }
```

\glxtrUpRho

```
12795 \newcommand*{\glxtrUpRho}{%
12796   \glshex 03C1% lower case rho
12797   \string=\glshex 03F1,% lower case rho variant
12798   \glshex 03A1% upper case rho
12799 }
```

\glxtrUpSigma

```
12800 \newcommand*{\glxtrUpSigma}{%
12801   \glshex 03C2% lower case sigma
12802   \string=\glshex 03C3,% lower case sigma
```

```
12803 \glshex 03A3% upper case sigma
12804 }
```

`\glxtrUpTau`

```
12805 \newcommand*{\glxtrUpTau}{%
12806 \glshex 03C4,% lower case tau
12807 \glshex 03A4% upper case tau
12808 }
```

`glxtrUpUpsilon`

```
12809 \newcommand*{\glxtrUpUpsilon}{%
12810 \glshex 03C5,% lower case upsilon
12811 \glshex 03A5% upper case upsilon
12812 }
```

`\glxtrUpPhi`

```
12813 \newcommand*{\glxtrUpPhi}{%
12814 \glshex 03C6% lower case phi
12815 \string=\glshex 03D5,% lower case phi variant
12816 \glshex 03A6% upper case phi
12817 }
```

`\glxtrUpChi`

```
12818 \newcommand*{\glxtrUpChi}{%
12819 \glshex 03C7,% lower case chi
12820 \glshex 03A7% upper case chi
12821 }
```

`\glxtrUpPsi`

```
12822 \newcommand*{\glxtrUpPsi}{%
12823 \glshex 03C8,% lower case psi
12824 \glshex 03A8% upper case psi
12825 }
```

`\glxtrUpOmega`

```
12826 \newcommand*{\glxtrUpOmega}{%
12827 \glshex 03C9,% lower case omega
12828 \glshex 03A9% upper case omega
12829 }
```

`MathItalicAlpha`

```
12830 \newcommand*{\glxtrMathItalicAlpha}{%
12831 \glshex 1D6FC,% lower case alpha (maths italic)
12832 \glshex 1D6E2% upper case alpha (maths italic)
12833 }
```

`rMathItalicBeta`

```
12834 \newcommand*{\glxtrMathItalicBeta}{%
```

```

12835 \glshex 1D6FD,% lower case beta (maths italic)
12836 \glshex 1D6E3% upper case beta (maths italic)
12837 }

```

#### MathItalicGamma

```

12838 \newcommand*{\glxtrMathItalicGamma}{%
12839 \glshex 1D6FE,% lower case gamma (maths italic)
12840 \glshex 1D6E4% upper case gamma (maths italic)
12841 }

```

#### MathItalicDelta

```

12842 \newcommand*{\glxtrMathItalicDelta}{%
12843 \glshex 1D6FF,% lower case delta (maths italic)
12844 \glshex 1D6E5% upper case delta (maths italic)
12845 }

```

#### MathItalicEpsilon

```

12846 \newcommand*{\glxtrMathItalicEpsilon}{%
12847 \glshex 1D700% lower case epsilon (maths italic)
12848 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
12849 \glshex 1D6E6% upper case epsilon (maths italic)
12850 }

```

#### MathItalicZeta

```

12851 \newcommand*{\glxtrMathItalicZeta}{%
12852 \glshex 1D701,% lower case zeta (maths italic)
12853 \glshex 1D6E7% upper case zeta (maths italic)
12854 }

```

#### MathItalicEta

```

12855 \newcommand*{\glxtrMathItalicEta}{%
12856 \glshex 1D702,% lower case eta (maths italic)
12857 \glshex 1D6E8% upper case eta (maths italic)
12858 }

```

#### MathItalicTheta

```

12859 \newcommand*{\glxtrMathItalicTheta}{%
12860 \glshex 1D703% lower case theta (maths italic)
12861 \string=\glshex 1D717,% lower case theta variant (maths italic)
12862 \glshex 1D6E9% upper case theta (maths italic)
12863 \string=\glshex 1D6F3% upper case theta variant (maths italic)
12864 }

```

#### MathItalicIota

```

12865 \newcommand*{\glxtrMathItalicIota}{%
12866 \glshex 1D704,% lower case iota (maths italic)
12867 \glshex 1D6EA% upper case iota (maths italic)
12868 }

```



MathItalicKappa

```
12869 \newcommand*{\glxtrMathItalicKappa}{%
12870   \glshex 1D705% lower case kappa (maths italic)
12871   \string=\glshex 1D718,% lower case kappa variant (maths italic)
12872   \glshex 1D6EB% upper case kappa (maths italic)
12873 }
```

athItalicLambda

```
12874 \newcommand*{\glxtrMathItalicLambda}{%
12875   \glshex 1D706,% lower case lambda (maths italic)
12876   \glshex 1D6EC% upper case lambda (maths italic)
12877 }
```

xtrMathItalicMu

```
12878 \newcommand*{\glxtrMathItalicMu}{%
12879   \glshex 1D707,% lower case mu (maths italic)
12880   \glshex 1D6ED% upper case mu (maths italic)
12881 }
```

xtrMathItalicNu

```
12882 \newcommand*{\glxtrMathItalicNu}{%
12883   \glshex 1D708,% lower case nu (maths italic)
12884   \glshex 1D6EE% upper case nu (maths italic)
12885 }
```

xtrMathItalicXi

```
12886 \newcommand*{\glxtrMathItalicXi}{%
12887   \glshex 1D709,% lower case xi (maths italic)
12888   \glshex 1D6EF% upper case xi (maths italic)
12889 }
```

thItalicOmicron

```
12890 \newcommand*{\glxtrMathItalicOmicron}{%
12891   \glshex 1D70A,% lower case omicron (maths italic)
12892   \glshex 1D6F0% upper case omicron (maths italic)
12893 }
```

xtrMathItalicPi

```
12894 \newcommand*{\glxtrMathItalicPi}{%
12895   \glshex 1D70B% lower case pi (maths italic)
12896   \string=\glshex 1D71B,% lower case pi variant (maths italic)
12897   \glshex 1D6F1% upper case pi (maths italic)
12898 }
```

trMathItalicRho

```
12899 \newcommand*{\glxtrMathItalicRho}{%
12900   \glshex 1D70C% lower case rho (maths italic)
12901   \string=\glshex 1D71A,% lower case rho variant (maths italic)
```

```

12902 \glshex 1D6F2% upper case rho (maths italic)
12903 }

```

MathItalicSigma

```

12904 \newcommand*{\glsxtrMathItalicSigma}{%
12905 \glshex 1D70D% lower case final sigma (maths italic)
12906 \string=\glshex 1D70E,% lower case sigma (maths italic)
12907 \glshex 1D6F4% upper case sigma (maths italic)
12908 }

```

trMathItalicTau

```

12909 \newcommand*{\glsxtrMathItalicTau}{%
12910 \glshex 1D70F,% lower case tau (maths italic)
12911 \glshex 1D6F5% upper case tau (maths italic)
12912 }

```

thItalicUpsilon

```

12913 \newcommand*{\glsxtrMathItalicUpsilon}{%
12914 \glshex 1D710,% lower case upsilon (maths italic)
12915 \glshex 1D6F6% upper case upsilon (maths italic)
12916 }

```

trMathItalicPhi

```

12917 \newcommand*{\glsxtrMathItalicPhi}{%
12918 \glshex 1D711% lower case phi (maths italic)
12919 \string=\glshex 1D719,% lower case phi variant (maths italic)
12920 \glshex 1D6F7% upper case phi (maths italic)
12921 }

```

trMathItalicChi

```

12922 \newcommand*{\glsxtrMathItalicChi}{%
12923 \glshex 1D712,% lower case chi (maths italic)
12924 \glshex 1D6F8% upper case chi (maths italic)
12925 }

```

trMathItalicPsi

```

12926 \newcommand*{\glsxtrMathItalicPsi}{%
12927 \glshex 1D713,% lower case psi (maths italic)
12928 \glshex 1D6F9% upper case psi (maths italic)
12929 }

```

MathItalicOmega

```

12930 \newcommand*{\glsxtrMathItalicOmega}{%
12931 \glshex 1D714,% lower case omega (maths italic)
12932 \glshex 1D6FA% upper case omega (maths italic)
12933 }

```

thItalicPartial

```
12934 \newcommand*{\glxtrMathItalicPartial}{%
12935   \glshex 1D715% partial differential (maths italic)
12936 }
```

MathItalicNabla

```
12937 \newcommand*{\glxtrMathItalicNabla}{%
12938   \glshex 1D6FB% nabla (maths italic)
12939 }
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
12940 \newcommand*{\glxtrdigitrules}{%
12941   0\string=\glshex 2080\string=\glshex 2070
12942   \string<1\string=\glshex 2081\string=\glshex 00B9
12943   \string<2\string=\glshex 2082\string=\glshex 00B2
12944   \string<3\string=\glshex 2083\string=\glshex 00B3
12945   \string<4\string=\glshex 2084\string=\glshex 2074
12946   \string<5\string=\glshex 2085\string=\glshex 2075
12947   \string<6\string=\glshex 2086\string=\glshex 2076
12948   \string<7\string=\glshex 2087\string=\glshex 2077
12949   \string<8\string=\glshex 2088\string=\glshex 2078
12950   \string<9\string=\glshex 2089\string=\glshex 2079
12951 }
```

BasicDigitrules Digits from the Basic Latin set.

```
12952 \newcommand*{\glxtrBasicDigitrules}{%
12953   0\string<1\string<2\string<3\string<4%
12954   \string<5\string<6\string<7\string<8\string<9%
12955 }
```

criptDigitrules Subscript digits.

```
12956 \newcommand*{\glxtrSubScriptDigitrules}{%
12957   \glshex 2080% subscript 0
12958   \string<\glshex 2081% subscript 1
12959   \string<\glshex 2082% subscript 2
12960   \string<\glshex 2083% subscript 3
12961   \string<\glshex 2084% subscript 4
12962   \string<\glshex 2085% subscript 5
12963   \string<\glshex 2086% subscript 6
12964   \string<\glshex 2087% subscript 7
12965   \string<\glshex 2088% subscript 8
12966   \string<\glshex 2089% subscript 9
12967 }
```

criptDigitrules Superscript digits.

```
12968 \newcommand*{\glxtrSuperScriptDigitrules}{%
12969   \glshex 2070% superscript 0
12970   \string<\glshex 00B9% superscript 1
```

```

12971 \string<\glshex 00B2% superscript 2
12972 \string<\glshex 00B3% superscript 3
12973 \string<\glshex 2074% superscript 4
12974 \string<\glshex 2075% superscript 5
12975 \string<\glshex 2076% superscript 6
12976 \string<\glshex 2077% superscript 7
12977 \string<\glshex 2078% superscript 8
12978 \string<\glshex 2079% superscript 9
12979 }

```

trfractionrules Vulgar fractions.

```

12980 \newcommand*{\glxtrfractionrules}{%
12981 \glshex 215F% fraction numerator one (1/)
12982 \string<\glshex 2189% zero thirds (0/3 = 0)
12983 \string<\glshex 2152% one tenth (1/10 = 0.1)
12984 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
12985 \string<\glshex 215B% one eighth (1/8 = 0.125)
12986 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
12987 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
12988 \string<\glshex 2155% one fifth (1/5 = 0.2)
12989 \string<\glshex 00BC% one quarter (1/4 = 0.25)
12990 \string<\glshex 2153% one third (1/3 ~ 0.333)
12991 \string<\glshex 215C% three eighths (3/8 = 0.375)
12992 \string<\glshex 2156% two fifths (2/5 = 0.4)
12993 \string<\glshex 00BD% one half (1/2 = 0.5)
12994 \string<\glshex 2157% three fifths (3/5 = 0.6)
12995 \string<\glshex 215D% five eighths (5/8 = 0.625)
12996 \string<\glshex 2154% two thirds (2/3 ~ 0.667)
12997 \string<\glshex 00BE% three quarters (3/4 = 0.75)
12998 \string<\glshex 2158% four fifths (4/5 = 0.8)
12999 \string<\glshex 215A% five sixths (5/6 ~ 0.833)
13000 \string<\glshex 215E% seven eighths (7/8 = 0.875)
13001 }

```

sxtrdialecthook Check for scripts associated with the document dialects.

```

13002 \renewcommand{\@glxtrdialecthook}{%
13003 \ifundef\CurrentTrackedScript
13004 {%
13005 \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
13006 {%
13007 \edef\CurrentTrackedScript{%
13008 \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
13009 }%
13010 {}%
13011 }%
13012 {}%
13013 \ifdef\CurrentTrackedScript
13014 {%
13015 \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix

```

```

13016 \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
13017 \let\CurrentTrackedTag\CurrentTrackedScript
13018 \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
13019 {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
13020 {}%
13021 \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
13022 }%
13023 {}%
13024 }

```

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.

```

13025 \ifdef\glsxtr@loaddialect
13026 {%
13027 \@ifpackageloaded{tracklang}
13028 {%
13029 \AnyTrackedLanguages
13030 {%
13031 \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
13032 }%
13033 {}%
13034 }
13035 {}
13036 }
13037 {}

```

## 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:

```
13038 \NeedsTeXFormat{LaTeX2e}
13039 \ProvidesPackage{glossaries-extra-stylemods}[2018/07/29 v1.34 (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`

```
13040 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
13041 \DeclareOption{all}{%
13042   \appto\@glxtr@loadstyles{%
13043     \RequirePackage{glossary-inline}%
13044     \RequirePackage{glossary-list}%
13045     \RequirePackage{glossary-tree}%
13046     \RequirePackage{glossary-mcols}%
13047     \RequirePackage{glossary-long}%
13048     \RequirePackage{glossary-longragged}%
13049     \RequirePackage{glossary-longbooktabs}%
13050     \RequirePackage{glossary-super}%
13051     \RequirePackage{glossary-superragged}%
13052     \RequirePackage{glossary-bookindex}%
13053   }
13054 }

13055 \DeclareOption*{%
13056   \IfFileExists{glossary-\CurrentOption.sty}
13057   {\eappto\@glxtr@loadstyles{%
13058     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
13059   }%
13060   {%
13061     \PackageError{glossaries-extra-styles}%

```

```

13062     {Unknown option ‘\CurrentOption’}{}%
13063   }%
13064 }

```

Process the package options:

```
13065 \ProcessOptions
```

Load the required packages:

```
13066 \@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`.

```
13067 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

`ewglossarystyle`

```

13068 \providecommand{\renewglossarystyle}[2]{%
13069   \ifcsundef{@glsstyle@#1}%
13070   {%
13071     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
13072   }%
13073   {%
13074     \csdef{@glsstyle@#1}{#2}%
13075   }%
13076 }

```

## 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.

```

13077 \ifdef{@glsstyle@listdotted}
13078 {%
13079   \renewglossarystyle{listdotted}{%
13080     \setglossarystyle{list}%
13081     \renewcommand*{\glossentry}[2]{%
13082       \item[]\makebox[\glslistdottedwidth][l]{%
13083         \glsentryitem{##1}%
13084         \glstarget{##1}{\glossentryname{##1}}%
13085         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13086         \glossentrydesc{##1}\glspostdescription}%
13087     \renewcommand*{\subglossentry}[3]{%
13088       \item[]\makebox[\glslistdottedwidth][l]{%
13089         \glssubentryitem{##2}%
13090         \glstarget{##2}{\glossentryname{##2}}%
13091         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
13092         \glossentrydesc{##2}\glspostdescription}%
13093   }

```

```
13094 }
13095 {%
```

Assume the style isn't required if it hasn't already been defined.

```
13096 }
```

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.

```
13097 \ifdef{\@glsstyle@list}
13098 {%
```

listprelocation Space before number list for top-level entries.

```
13099 \newcommand{\glslistprelocation}{\glstrprelocation}
```

childprelocation Space before number list for child entries.

```
13100 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
13101 \newcommand{\glslistchildpostlocation}{.}
```

\glslistdesc

```
13102 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

Redefine list to use these commands.

```
13103 \renewglossarystyle{list}{%
13104   \renewenvironment{theglossary}%
13105     {\begin{description}}{\end{description}}%
13106   \renewcommand*\{glossaryheader}{}%
13107   \renewcommand*\{glsgroupheading}[1]{}%
13108   \renewcommand*\{glossentry}[2]{%
13109     \item[\glssentryitem{##1}%
13110       \glstarget{##1}{\glossentryname{##1}}]
13111     \glslistdesc{##1}\glslistprelocation ##2}%
13112   \renewcommand*\{subglossentry}[3]{%
13113     \glssubentryitem{##2}%
13114     \glstarget{##2}{\strut}\space
13115     \glslistdesc{##2}%
13116     \glslistchildprelocation ##3\glslistchildpostlocation}%
13117   \renewcommand*\{glsgroupskip}{\ifglsgroupskip\else\indexspace\fi}%
13118 }
13119 }
13120 }
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
13121 \ifdef{\@glsstyle@altlist}
13122 {%
13123   \renewglossarystyle{altlist}{%
```



```

13124 \setglossarystyle{list}%
13125 \renewcommand*{\glossentry}[2]{%
13126   \item[\glentryitem{##1}%
13127     \glstarget{##1}{\glossentryname{##1}}]%
13128   \mbox{}\par\nobreak\@afterheading
13129   \glslistdesc{##1}\glslistprelocation ##2}%
13130 \renewcommand{\subglossentry}[3]{%
13131   \par
13132   \glssubentryitem{##2}%
13133   \glstarget{##2}{\strut}\glslistdesc{##2}%
13134   \glslistchildprelocation ##3}%
13135 }
13136 }
13137 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

13138 \ifdef{\@glsstyle@listgroup}
13139 {%
13140   \renewglossarystyle{listgroup}{%
13141     \setglossarystyle{list}%
13142     \renewcommand*{\glsgroupheading}[1]{%
13143       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
13144       \mbox{}\par\nobreak\@afterheading
13145     }%
13146   }
13147 }
13148 {}

```

Similarly for listhypergroup.

```

13149 \ifdef{\@glsstyle@listhypergroup}
13150 {%
13151   \renewglossarystyle{listhypergroup}{%
13152     \setglossarystyle{list}%
13153     \renewcommand*{\glossaryheader}{%
13154       \glslistnavigationitem{\glsnavigation}}%
13155     \renewcommand*{\glsgroupheading}[1]{%
13156       \item[\glslistgroupheaderfmt
13157         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13158       \mbox{}\par\nobreak\@afterheading
13159     }%
13160   }
13161 }
13162 {}

```

Similarly for altlistgroup.

```

13163 \ifdef{\@glsstyle@altlistgroup}
13164 {%
13165   \renewglossarystyle{altlistgroup}{%
13166     \setglossarystyle{altlist}%
13167     \renewcommand*{\glsgroupheading}[1]{%
13168       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%

```

```

13169      \mbox{}\par\nobreak\@afterheading
13170    }%
13171  }
13172 }
13173 {}

```

Similarly for altlisthypergroup.

```

13174 \ifdef{\@glsstyle@altlisthypergroup}
13175 {%
13176   \renewglossarystyle{altlisthypergroup}{%
13177     \setglossarystyle{altlist}%
13178     \renewcommand*\glossaryheader{%
13179       \glslistnavigationitem{\glsnavigation}}%
13180     \renewcommand*\glsgroupheading[1]{%
13181       \item[\glslistgroupheaderfmt
13182         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
13183       \mbox{}\par\nobreak\@afterheading
13184     }%
13185   }
13186 }
13187 {}

```

## 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`.

```

13188 \ifcsdef{@glsstyle@long}
13189 {%
13190   \renewglossarystyle{long}{%
13191     \renewenvironment{theglossary}%
13192       {\begin{longtable}[lp{\glsdescwidth}]}%
13193       {\end{longtable}}%
13194     \renewcommand*\glossaryheader{}%
13195     \renewcommand*\glsgroupheading[1]{}%
13196     \renewcommand{\glossentry}[2]{%
13197       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13198       \glossentrydesc{##1}\glspostdescription
13199       \glxtrprelocation ##2\tabularnewline
13200     }%
13201     \renewcommand{\subglossentry}[3]{%
13202       &
13203       \glssubentryitem{##2}%
13204       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13205       \glxtrprelocation ##3\tabularnewline
13206     }%
13207     \ifglsnogroupskip
13208       \renewcommand*\glsgroupskip{}%
13209     \else

```

```

13210      \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
13211      \fi
13212    }
13213  }
13214 {}

```

Three column style:

```

13215 \ifcsdef{@glsstyle@long3col}
13216 {%
13217   \renewglossarystyle{long3col}{%
13218     \renewenvironment{theglossary}%
13219       {\begin{longtable}\lp{\glsdescwidth}p{\glspagelistwidth}}}%
13220     {\end{longtable}}}%
13221   \renewcommand*{\glossaryheader}{}%
13222   \renewcommand*{\glsgroupheading}[1]{}%
13223   \renewcommand{\glossentry}[2]{%
13224     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13225     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13226   }%
13227   \renewcommand{\subglossentry}[3]{%
13228     &
13229     \glssubentryitem{##2}%
13230     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13231     ##3\tabularnewline
13232   }%

```

Conditional needs to be outside of \glsgroupskip otherwise it can cause “Incomplete \iftrue” errors.

```

13233   \ifglsnogroupskip
13234     \renewcommand*{\glsgroupskip}{}%
13235   \else
13236     \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13237   \fi
13238 }
13239 }
13240 {}

```

Four column style:

```

13241 \ifcsdef{@glsstyle@long4col}
13242 {%
13243   \renewglossarystyle{long4col}{%
13244     \renewenvironment{theglossary}%
13245       {\begin{longtable}{llll}}}%
13246     {\end{longtable}}}%
13247   \renewcommand*{\glossaryheader}{}%
13248   \renewcommand*{\glsgroupheading}[1]{}%
13249   \renewcommand{\glossentry}[2]{%
13250     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13251     \glossentrydesc{##1}\glspostdescription &
13252     \glossentrysymbol{##1} &
13253     ##2\tabularnewline

```

```

13254 }%
13255 \renewcommand{\subglossentry}[3]{%
13256     &
13257     \glssubentryitem{##2}%
13258     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13259     \glossentrysymbol{##2} & ##3\tabularnewline
13260 }%

13261 \ifglsgroupskip
13262     \renewcommand*\{glsgroupskip}{}%
13263 \else
13264     \renewcommand*\{glsgroupskip}{& & \tabularnewline}%
13265 \fi
13266 }
13267 }
13268 {}

```

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`.

```

13269 \ifcsdef{@glstyle@longragged}
13270 {%
13271     \renewglossarystyle{longragged}{%
13272         \renewenvironment{theglossary}%
13273             {\begin{longtable}[l>\raggedright]p{\glstdescwidth}}}%
13274             {\end{longtable}}}%
13275     \renewcommand*\{glossaryheader}{}%
13276     \renewcommand*\{glsgroupheading}[1]{}%
13277     \renewcommand{\glossentry}[2]{%
13278         \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13279         \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
13280         \tabularnewline
13281     }%
13282     \renewcommand{\subglossentry}[3]{%
13283         &
13284         \glssubentryitem{##2}%
13285         \glstarget{##2}{\strut}\glossentrydesc{##2}%
13286         \glspostdescription\glxtrprelocation ##3%
13287         \tabularnewline
13288     }%
13289     \ifglsgroupskip
13290         \renewcommand*\{glsgroupskip}{}%
13291     \else
13292         \renewcommand*\{glsgroupskip}{& \tabularnewline}%

```

```

13293 \fi
13294 }
13295 }
13296 {}

```

Three and four column styles don't use `\glxtrprelocation` since the number list is in its own column.

```

13297 \ifcsdef{@glsstyle@longragged3col}
13298 {%
13299 \renewglossarystyle{longragged3col}{%
13300 \renewenvironment{theglossary}%
13301 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
13302 >{\raggedright}p{\glspagelistwidth}}}%
13303 {\end{longtable}}}%
13304 \renewcommand*{\glossaryheader}{}%
13305 \renewcommand*{\glsgroupheading}[1]{}%
13306 \renewcommand{\glossentry}[2]{%
13307 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13308 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
13309 }%
13310 \renewcommand{\subglossentry}[3]{%
13311 &
13312 \glssubentryitem{##2}%
13313 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13314 ##3\tabularnewline
13315 }%
13316 \ifglsgroupskip
13317 \renewcommand*{\glsgroupskip}{}%
13318 \else
13319 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13320 \fi
13321 }
13322 }
13323 {}

```

Four column style:

```

13324 \ifcsdef{@glsstyle@altlongragged4col}
13325 {%
13326 \renewglossarystyle{altlongragged4col}{%
13327 \renewenvironment{theglossary}%
13328 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
13329 >{\raggedright}p{\glspagelistwidth}}}%
13330 {\end{longtable}}}%
13331 \renewcommand*{\glossaryheader}{}%
13332 \renewcommand*{\glsgroupheading}[1]{}%
13333 \renewcommand{\glossentry}[2]{%
13334 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13335 \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
13336 ##2\tabularnewline

```

```

13337 }%
13338 \renewcommand{\subglossentry}[3]{%
13339     &
13340     \glssubentryitem{##2}%
13341     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13342     \glossentrysymbol{##2} & ##3\tabularnewline
13343 }%
13344 \ifglsgnogroupskip
13345     \renewcommand*{\glsgroupskip}{}%
13346 \else
13347     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13348 \fi
13349 }
13350 }
13351 {}

```

## 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`.

```

13352 \ifcsdef{@glstyle@super}
13353 {%
13354     \renewglossarystyle{super}{%
13355         \renewenvironment{theglossary}%
13356             {\tablehead{}}\tabletail{}}%
13357         \begin{supertabular}{lp{\glsdescwidth}}%
13358             {\end{supertabular}}%
13359         \renewcommand*{\glossaryheader}{}%
13360         \renewcommand*{\glsgroupheading}[1]{}%
13361         \renewcommand{\glossentry}[2]{%
13362             \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13363             \glossentrydesc{##1}\glspostdescription
13364             \glstrprelocation ##2\tabularnewline
13365         }%
13366         \renewcommand{\subglossentry}[3]{%
13367             &
13368             \glssubentryitem{##2}%
13369             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13370             \glstrprelocation ##3\tabularnewline
13371         }%
13372         \ifglsgnogroupskip
13373             \renewcommand*{\glsgroupskip}{}%
13374         \else
13375             \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13376         \fi
13377     }
13378 }
13379 {}

```

Three column style:

```

13380 \ifcsdef{@glsstyle@super3col}
13381 {%
13382   \renewglossarystyle{super3col}{%
13383     \renewenvironment{theglossary}%
13384       {\tablehead{}\tabletail{}}%
13385       \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
13386       {\end{supertabular}}}%
13387   \renewcommand*{\glossaryheader}{}%
13388   \renewcommand*{\glsgroupheading}[1]{}%
13389   \renewcommand{\glossentry}[2]{%
13390     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13391     \glsentrydesc{##1}\glspostdescription & ##2\tabularnewline
13392   }%
13393   \renewcommand{\subglossentry}[3]{%
13394     &
13395     \glssubentryitem{##2}%
13396     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13397     ##3\tabularnewline
13398   }%
13399   \ifglsgroupskip
13400     \renewcommand*{\glsgroupskip}{}%
13401   \else
13402     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13403   \fi
13404 }
13405 }
13406 {}

```

Four column styles:

```

13407 \ifcsdef{@glsstyle@super4col}
13408 {%
13409   \renewglossarystyle{super4col}{%
13410     \renewenvironment{theglossary}%
13411       {\tablehead{}\tabletail{}}%
13412       \begin{supertabular}{llll}}}%
13413       \end{supertabular}}}%
13414   \renewcommand*{\glossaryheader}{}%
13415   \renewcommand*{\glsgroupheading}[1]{}%
13416   \renewcommand{\glossentry}[2]{%
13417     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
13418     \glsentrydesc{##1}\glspostdescription &
13419     \glsentrysymbol{##1} & ##2\tabularnewline
13420   }%
13421   \renewcommand{\subglossentry}[3]{%
13422     &
13423     \glssubentryitem{##2}%
13424     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
13425     \glsentrysymbol{##2} & ##3\tabularnewline

```

```

13426 }%
13427 \ifglsgroupskip
13428 \renewcommand*{\glsgroupskip}{}%
13429 \else
13430 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13431 \fi
13432 }
13433 }
13434 {}

```

## 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`.

```

13435 \ifcsdef{@glstyle@superragged}
13436 {%
13437 \renewglossarystyle{superragged}{%
13438 \renewenvironment{theglossary}%
13439 {\tablehead{}\tabletail}%
13440 \begin{supertabular}{l>{\raggedright}p{\glsglwidth}}%
13441 {\end{supertabular}}%
13442 \renewcommand*{\glossaryheader}{}%
13443 \renewcommand*{\glsgroupheading}[1]{}%
13444 \renewcommand{\glossentry}[2]{%
13445 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13446 \glossentrydesc{##1}\glspostdescription\glxstrprelocation ##2%
13447 \tabularnewline
13448 }%
13449 \renewcommand{\subglossentry}[3]{%
13450 &
13451 \glssubentryitem{##2}%
13452 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
13453 \glxstrprelocation ##3%
13454 \tabularnewline
13455 }%
13456 \ifglsgroupskip
13457 \renewcommand*{\glsgroupskip}{}%
13458 \else
13459 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13460 \fi
13461 }
13462 }
13463 {}

```

Three column style:

```

13464 \ifcsdef{@glstyle@superragged3col}
13465 {%

```



```

13466 \renewglossarystyle{superragged3col}{%
13467   \renewenvironment{theglossary}%
13468     {\tablehead{}\tabletail{}}%
13469     \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
13470       >{\raggedright}p{\glspagelistwidth}}}%
13471     {\end{supertabular}}%
13472   \renewcommand*{\glossaryheader}{}%
13473   \renewcommand*{\glsgroupheading}[1]{}%
13474   \renewcommand{\glossentry}[2]{%
13475     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13476     \glossentrydesc{##1}\glspostdescription &
13477     ##2\tabularnewline
13478   }%
13479   \renewcommand{\subglossentry}[3]{%
13480     &
13481     \glssubentryitem{##2}%
13482     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13483     ##3\tabularnewline
13484   }%

13485   \ifglsgroupskip
13486     \renewcommand*{\glsgroupskip}{}%
13487   \else
13488     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
13489   \fi
13490 }
13491 }
13492 {}

```

Four columns:

```

13493 \ifcsdef{@glsstyle@altsuperragged4col}
13494 {%
13495   \renewglossarystyle{altsuperragged4col}{%
13496     \renewenvironment{theglossary}%
13497       {\tablehead{}\tabletail{}}%
13498       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}1%
13499         >{\raggedright}p{\glspagelistwidth}}}%
13500       {\end{supertabular}}%
13501     \renewcommand*{\glossaryheader}{}%
13502     \renewcommand{\glossentry}[2]{%
13503       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13504       \glossentrydesc{##1}\glspostdescription &
13505       \glossentrysymbol{##1} & ##2\tabularnewline
13506     }%
13507     \renewcommand{\subglossentry}[3]{%
13508       &
13509       \glssubentryitem{##2}%
13510       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13511       \glossentrysymbol{##2} & ##3\tabularnewline
13512     }%

```

```

13513 \ifglsnogroupskip
13514 \renewcommand*{\glsgroupskip}{}%
13515 \else
13516 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13517 \fi
13518 }
13519 }
13520 {}

```

## 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`.

```

13521 \ifdef{\@glsstyle@inline}
13522 {%
13523 \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
13524 \renewcommand*{\glsinlinedescformat}[3]{%
13525 \space#1\glxtrpostdescription}
13526 \renewcommand*{\glsinlinesubdescformat}[3]{%
13527 #1\glxtrpostdescription}

```

Just use `\glxtrpostdescription` instead of `\glspostdescription`.

```

13528 }
13529 {}

```

## 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.

```

13530 \ifdef\glstreenamfmt
13531 {

```

```

\glstreedefaultnamfmt

```

```

13532 \newcommand{\glstreedefaultnamfmt}[1]{\textbf{#1}}

```

```

\glstreenamfmt

```

```

13533 \renewcommand{\glstreenamfmt}[1]{\glstreedefaultnamfmt{#1}}

```

`\glstreegroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```

13534 \def\glstreegroupheaderfmt#1{\glstreedefaultnamfmt{#1}}

```

reenavigationfmt This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
13535 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
13536 }
13537 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
13538 \ifdef{\@glsstyle@index}
13539 {
```

treeprelocation The space before the number list for top-level entries. This is shared by the other tree styles.

```
13540 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

childprelocation The space before the number list for child entries. This is shared by the other tree styles.

```
13541 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Modify the index style.

```
13542 \renewglossarystyle{index}{%
13543   \renewenvironment{theglossary}%
13544     {\setlength{\parindent}{0pt}%
13545     \setlength{\parskip}{0pt plus 0.3pt}%
13546     \let\item\glstreeitem
13547     \let\subitem\glstreesubitem
13548     \let\subsubitem\glstreesubsubitem
13549   }%
13550   {\par}%
13551   \renewcommand*{\glossaryheader}{}%
13552   \renewcommand*{\glsgroupheading}[1]{}%
13553   \renewcommand*{\glossentry}[2]{}%
13554     \item\glssentryitem{##1}%
13555     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13556     \glstreesymbol{##1}%
13557     \glstreedesc{##1}%
13558     \glstreeprelocation ##2%
13559   }%
13560   \renewcommand{\subglossentry}[3]{}%
13561     \ifcase##1\relax
13562       \item
13563     \or
13564       \subitem
13565       \glssubentryitem{##2}%
13566     \else
13567       \subsubitem
13568     \fi
13569     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13570     \glstreechildsymbol{##2}%
13571     \glstreechilddesc{##2}%
13572     \glstreechildprelocation ##3%
13573   }%
```

```

13574 \renewcommand*{\glsgroupskip}{\ifglsgnigroupskip\else\indexspace\fi}%
13575 }
13576 }
13577 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

13578 \ifdef{\@glsstyle@indexgroup}
13579 {%
13580 \renewglossarystyle{indexgroup}{%
13581 \setglossarystyle{index}%
13582 \renewcommand*{\glsgroupheading}[1]{%
13583 \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
13584 \nopagebreak\indexspace
13585 \nobreak\@afterheading
13586 }%
13587 }
13588 }
13589 {}

```

Similarly for indexhypergroup.

```

13590 \ifdef{\@glsstyle@indexhypergroup}
13591 {%
13592 \renewglossarystyle{indexhypergroup}{%
13593 \setglossarystyle{index}%
13594 \renewcommand*{\glossaryheader}{%
13595 \item\glstreenavigationfmt{\glsnavigation}%
13596 \nobreak\@afterheading\indexspace}%
13597 \renewcommand*{\glsgroupheading}[1]{%
13598 \item\glstreegroupheaderfmt
13599 {\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13600 \nopagebreak\indexspace
13601 \nobreak\@afterheading}%
13602 }%
13603 }
13604 {}

```

Adjust tree style to remove hard coded space before number list.

```

13605 \ifdef{\@glsstyle@tree}
13606 {%
13607 %Provide a command for use with the \glostyle{tree} styles that displays
13608 %the pre-description separator, the
13609 %description and post-description hook.
13610 %\begin{macro}{\glstreedesc}
13611 %\changes{1.31}{2018-05-09}{new}
13612 % \begin{macrocode}
13613 \newcommand{\glstreedesc}[1]{%
13614 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13615 }

```

Similarly for the symbol.

`\glstreesymbol`

```
13616 \newcommand{\glstreesymbol}[1]{%
13617   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
13618 }%
```

And for the child entries:

`lstreechilddesc`

```
13619 \newcommand{\glstreechilddesc}[1]{%
13620   \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
13621 }%
```

`treechildsymbol` This just behaves in the same way as the top-level.

```
13622 \newcommand{\glstreechildsymbol}[1]{%
13623   \glstreesymbol{#1}%
13624 }%

13625 \renewglossarystyle{tree}{%
13626   \renewenvironment{theglossary}%
13627     {\setlength{\parindent}{0pt}%
13628     \setlength{\parskip}{0pt plus 0.3pt}}%
13629   {%
13630     \renewcommand*{\glossaryheader}{}%
13631     \renewcommand*{\glsgroupheading}[1]{}%
13632     \renewcommand{\glossentry}[2]{%
13633       \hangindent0pt\relax
13634       \parindent0pt\relax
13635       \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13636       \glstreesymbol{##1}%
13637       \glstreedesc{##1}%
13638       \glstreeprelocation##2\par
13639     }%
13640     \renewcommand{\subglossentry}[3]{%
13641       \hangindent##1\glstreeindent\relax
13642       \parindent##1\glstreeindent\relax
13643       \ifnum##1=1\relax
13644         \glssubentryitem{##2}%
13645       \fi
13646       \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13647       \glstreechildsymbol{##2}%
13648       \glstreechilddesc{##2}%
13649       \glstreechildprelocation ##3\par
13650     }%
13651     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13652   }%
13653 }
13654 {}
```

The `treegroup` style is redefined to discourage a page break after the heading.

```
13655 \ifdef{\@glstyle@treegroup}
```

```

13656 {%
13657   \renewglossarystyle{treegroup}{%
13658     \setglossarystyle{tree}%
13659     \renewcommand{\glsgroupheading}[1]{\par
13660       \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
13661       \nopagebreak\indexspace\nobreak\@afterheading}%
13662   }
13663 }
13664 {}

```

Similarly for treehypergroup

```

13665 \ifdef{\@glsstyle@treehypergroup}
13666 {%
13667   \renewglossarystyle{treehypergroup}{%
13668     \setglossarystyle{tree}%
13669     \renewcommand*\{\glossaryheader}{%
13670       \par\noindent\glstreenavigationfmt{\glstravigation}\par
13671       \nobreak\@afterheading\indexspace}%
13672     \renewcommand*\{\glsgroupheading}[1]{%
13673       \par\noindent
13674       \glstreegroupheaderfmt
13675       {\glstravigationtarget{##1}{\glsgrouptitle{##1}}}\par
13676       \nopagebreak\indexspace\nobreak\@afterheading}%
13677   }
13678 }
13679 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

13680 \ifdef{\@glsstyle@treenoname}
13681 {%
13682   %Provide a command for use with the \glsstyle{treenoname} styles that displays
13683   %the pre-description separator, the
13684   %description and post-description hook.
13685   %\begin{macro}{\glstreenonamedesc}
13686   %\changes{1.31}{2018-05-09}{new}
13687   %   \begin{macrocode}
13688   \newcommand{\glstreenonamedesc}[1]{%
13689     \glstreepredesc\glossentrydesc{#1}\glspostdescription
13690   }%

```

Similarly for the symbol.

treenonamesymbol

```

13691   \newcommand{\glstreenonamesymbol}[1]{%
13692     \ifglshassymbol{#1}{\space\glossentrysymbol{#1}}{}}%
13693   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

13694   \newcommand{\glstreenonamechilddesc}[1]{%
13695     \glossentrydesc{#1}\glspostdescription
13696   }%

```

```

13697 \renewglossarystyle{treenoname}{%
13698   \renewenvironment{theglossary}%
13699     {\setlength{\parindent}{0pt}%
13700      \setlength{\parskip}{0pt plus 0.3pt}}%
13701     {}%
13702   \renewcommand*{\glossaryheader}{}%
13703   \renewcommand*{\glsgroupheading}[1]{}%
13704   \renewcommand{\glossentry}[2]{%
13705     \hangindent0pt\relax
13706     \parindent0pt\relax
13707     \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13708     \glstreenonamesymbol{##1}%
13709     \glstreenonamedesc{##1}%
13710     \glstreeprelocation##2\par
13711   }%
13712   \renewcommand{\subglossentry}[3]{%
13713     \hangindent##1\glstreeindent\relax
13714     \parindent##1\glstreeindent\relax
13715     \ifnum##1=1\relax
13716       \glssubentryitem{##2}%
13717     \fi
13718     \glstarget{##2}{\strut}%
13719     \glstreenonamechilddesc{##2}%
13720     \glstreechildprelocation##3\par
13721   }%
13722   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13723 }
13724 }
13725 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

13726 \ifdef{\@glsstyle@treenonamegroup}
13727 {%
13728   \renewglossarystyle{treenonamegroup}{%
13729     \setglossarystyle{treenoname}%
13730     \renewcommand{\glsgroupheading}[1]{\par
13731       \noindent\glstreegroupheaderfmt
13732       {\glsgetgrouptitle{##1}}}%
13733     \nopagebreak\indexspace\nobreak\@afterheading
13734   }%
13735 }
13736 }
13737 {}

```

Similarly for `treenonamehypergroup`

```

13738 \ifdef{\@glsstyle@treenonamehypergroup}
13739 {%
13740   \renewglossarystyle{treenonamehypergroup}{%
13741     \setglossarystyle{treenoname}%
13742     \renewcommand*{\glossaryheader}{%

```

```

13743      \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13744      \nobreak\@afterheading\indexspace}%
13745      \renewcommand*{\glsgroupheading}[1]{%
13746      \par\noindent
13747      \glstreegroupheaderfmt
13748      {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13749      \nopagebreak\indexspace\nobreak\@afterheading}%
13750  }
13751 }
13752 {}

```

The `almtree` style is redefined to make it easier to make minor adjustments.

```

13753 \ifdef{\glstyle@almtree}
13754 {}%

```

Only redefine this if it's already been defined.

`SymbolDescLocation` `\glxtralttreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```

13755 \newcommand{\glxtralttreeSymbolDescLocation}[2]{%
13756   {%
13757     \let\par\glxtrAltTreePar
13758     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
13759     \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
13760   }%
13761 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

13762 \newlength\glxtrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

13763 \newcommand{\glxtrAltTreePar}{%
13764   \@@par
13765   \glxtrAltTreeSetHangIndent
13766   \setlength{\parindent}{\dimexpr\hangindent+\glxtrAltTreeIndent}%
13767 }

```

`SymbolDescLocation` `\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

13768 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
13769   \glxtralttreeSymbolDescLocation{#2}{#3}%
13770 }

```



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.

```
13771 \newlength\glxstrtreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
13772 \newcommand*\glxtralttreeInit{%
13773   \settowidth{\glxstrtreetopindent}{\glstreenamefmt{\glsggetwidestname\space}}%
13774   \glxstrAltTreeIndent=\parindent
13775 }
```

\gglsetwidest The original \glsetwidest only uses \def. This uses \gdef.

```
13776 \newcommand*\gglsetwidest}[2][0]{%
13777   \csgdef{@glswidestname\romannumeral#1}{#2}%
13778 }
```

\eglssetwidest The original \glsetwidest only uses \def. This uses \protected@csedef.

```
13779 \newcommand*\eglssetwidest}[2][0]{%
13780   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13781 }
```

\xglsetwidest Like the above but uses \protected@csxdef.

```
13782 \newcommand*\xglsetwidest}[2][0]{%
13783   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13784 }
```

glupdatewidest Only sets if new value is wider than old value.

```
13785 \newcommand*\glupdatewidest}[2][0]{%
13786   \ifcsundef{@glswidestname\romannumeral#1}%
13787   {\csdef{@glswidestname\romannumeral#1}{#2}}%
13788   {%
13789     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13790     \settowidth{\dimen@ii}{#2}%
13791     \ifdim\dimen@ii>\dimen@
13792     \csdef{@glswidestname\romannumeral#1}{#2}%
13793     \fi
13794   }%
13795 }
```

glupdatewidest As above but global definition.

```
13796 \newcommand*\gglupdatewidest}[2][0]{%
13797   \ifcsundef{@glswidestname\romannumeral#1}%
13798   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
13799   {%
13800     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13801     \settowidth{\dimen@ii}{#2}%
13802     \ifdim\dimen@ii>\dimen@
13803     \csgdef{@glswidestname\romannumeral#1}{#2}%
13804     \fi
```

```

13805     }%
13806   }

```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```

13807   \newcommand*{\eglsupdatewidest}[2][0]{%
13808     \ifcsundef{@glswidestname\romannumeral#1}%
13809     {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
13810     {%
13811       \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13812       \settowidth{\dimen@ii}{#2}%
13813       \ifdim\dimen@ii>\dimen0
13814         \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13815       \fi
13816     }%
13817   }

```

`glsupdatewidest` As above but global.

```

13818   \newcommand*{\xglsupdatewidest}[2][0]{%
13819     \ifcsundef{@glswidestname\romannumeral#1}%
13820     {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
13821     {%
13822       \settowidth{\dimen0}{\csuse{@glswidestname\romannumeral#1}}%
13823       \settowidth{\dimen@ii}{#2}%
13824       \ifdim\dimen@ii>\dimen0
13825         \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13826       \fi
13827     }%
13828   }

```

`glsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

13829   \newcommand*{\glsgetwidestname}{\@glswidestname}

```

`glsgetwidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

13830   \newcommand*{\glsgetwidestsubname}[1]{%
13831     \ifcsundef{@glswidestname\romannumeral#1}%
13832     {\@glswidestname}%
13833     {\csuse{@glswidestname\romannumeral#1}}%
13834   }

```

`glsfindwidesttoplevelname` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`.

```

13835   \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.

```

13836   \newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
13837     \dimen@=0pt\relax

```

```

13838 \gls@tmplen=Opt\relax
13839 \foralllglossaries[#1]{\@gls@type}%
13840 {%
13841 \forglsentries[\@gls@type]{\@glo@label}%
13842 {%
13843 \ifglsused{\@glo@label}%
13844 {%
13845 \ifglshasparent{\@glo@label}%
13846 {}%
13847 {%
13848 \settowidth{\dimen@}%
13849 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13850 \ifdim\dimen@>\gls@tmplen
13851 \gls@tmplen=\dimen@
13852 \eglssetwidest{\glsentryname{\@glo@label}}%
13853 \fi
13854 }%
13855 }%
13856 {}%
13857 }%
13858 }%
13859 }

```

**destUsedAnyName** Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

13860 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
13861 \dimen@=Opt\relax
13862 \gls@tmplen=Opt\relax
13863 \foralllglossaries[#1]{\@gls@type}%
13864 {%
13865 \forglsentries[\@gls@type]{\@glo@label}%
13866 {%
13867 \ifglsused{\@glo@label}%
13868 {%
13869 \settowidth{\dimen@}%
13870 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13871 \ifdim\dimen@>\gls@tmplen
13872 \gls@tmplen=\dimen@
13873 \eglssetwidest{\glsentryname{\@glo@label}}%
13874 \fi
13875 }%
13876 {}%
13877 }%
13878 }%
13879 }

```

**ndWidestAnyName** Like the above but doesn't check is the entry has been used.

```

13880 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
13881 \dimen@=Opt\relax

```

```

13882 \gls@tmplen=0pt\relax
13883 \foralllglossaries[#1]{\@gls@type}%
13884 {%
13885   \forallglsentries[\@gls@type]{\@glo@label}%
13886   {%
13887     \settowidth{\dimen@}%
13888     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13889     \ifdim\dimen@>\gls@tmplen
13890       \gls@tmplen=\dimen@
13891       \eglssetwidest{\glsentryname{\@glo@label}}%
13892     \fi
13893   }%
13894 }%
13895 }

```

estUsedLevelTwo This is like \glsFindWidestUsedTopLevelName but also sets the first two sub-levels as well.  
Any entry that has a great-grandparent is ignored.

```

13896 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
13897   \dimen@=0pt\relax
13898   \dimen@i=0pt\relax
13899   \dimen@ii=0pt\relax
13900   \foralllglossaries[#1]{\@gls@type}%
13901   {%
13902     \forallglsentries[\@gls@type]{\@glo@label}%
13903     {%
13904       \ifglsused{\@glo@label}%
13905       {%
13906         \ifglshasparent{\@glo@label}%
13907         {%
13908           \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}}@parent}}%
13909           \ifglshasparent{\@glo@parent}%
13910           {%
13911             \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}}@parent}}%
13912             \ifglshasparent{\@glo@parent}%
13913             {}%
13914           {%
13915             \settowidth{\gls@tmplen}%
13916             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13917             \ifdim\gls@tmplen>\dimen@ii
13918               \dimen@ii=\gls@tmplen
13919               \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13920             \fi
13921           }%
13922         }%
13923       {%
13924         \settowidth{\gls@tmplen}%
13925         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13926         \ifdim\gls@tmplen>\dimen@i
13927           \dimen@i=\gls@tmplen

```

```

13928         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13929     \fi
13930 }%
13931 }%
13932 {%
13933     \settowidth{\gls@tmplen}%
13934         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13935     \ifdim\gls@tmplen>\dimen@
13936         \dimen@=\gls@tmplen
13937         \eglssetwidest{\glsentryname{\@glo@label}}%
13938     \fi
13939 }%
13940 }%
13941 {}%
13942 }%
13943 }%
13944 }

```

**dWidestLevelTwo** This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

13945 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
13946     \dimen@=0pt\relax
13947     \dimen@i=0pt\relax
13948     \dimen@ii=0pt\relax
13949     \foralllglossaries[#1]{\@gls@type}%
13950     {%
13951         \forglsentries[\@gls@type]{\@glo@label}%
13952         {%
13953             \ifglshasparent{\@glo@label}%
13954             {%
13955                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
13956                 \ifglshasparent{\@glo@parent}%
13957                 {%
13958                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
13959                     \ifglshasparent{\@glo@parent}%
13960                     {}%
13961                 }%
13962                 \settowidth{\gls@tmplen}%
13963                     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13964                 \ifdim\gls@tmplen>\dimen@ii
13965                     \dimen@ii=\gls@tmplen
13966                     \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13967                 \fi
13968             }%
13969         }%
13970     }%
13971     \settowidth{\gls@tmplen}%
13972         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13973     \ifdim\gls@tmplen>\dimen@i
13974         \dimen@i=\gls@tmplen

```

```

13975         \eglssetwidest[1]{\glsentryname{\@glo@label}}}%
13976     \fi
13977 }%
13978 }%
13979 {%
13980     \settowidth{\gls@tmplen}%
13981     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
13982     \ifdim\gls@tmplen>\dimen@
13983         \dimen@=\gls@tmplen
13984         \eglssetwidest{\glsentryname{\@glo@label}}}%
13985     \fi
13986 }%
13987 }%
13988 }%
13989 }

```

**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.

```

13990 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
13991     \dimen@=0pt\relax
13992     \gls@tmplen=0pt\relax
13993     #2=0pt\relax
13994     \foralllglossaries[#1]{\@gls@type}%
13995     {%
13996         \forglsentries[\@gls@type]{\@glo@label}%
13997         {%
13998             \ifglsused{\@glo@label}%
13999             {%
14000                 \settowidth{\dimen@}%
14001                 {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
14002                 \ifdim\dimen@>\gls@tmplen
14003                     \gls@tmplen=\dimen@
14004                     \eglssetwidest{\glsentryname{\@glo@label}}}%
14005                 \fi
14006                 \settowidth{\dimen@}%
14007                 {\glsentrysymbol{\@glo@label}}}%
14008                 \ifdim\dimen@>#2\relax
14009                     #2=\dimen@
14010                 \fi
14011             }%
14012         }%
14013     }%
14014 }%
14015 }

```

**stAnyNameSymbol** Like the above but doesn't check if the entry has been used.

```

14016 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
14017     \dimen@=0pt\relax
14018     \gls@tmplen=0pt\relax

```

```

14019      #2=Opt\relax
14020      \foralllglossaries[#1]{\@gls@type}%
14021      {%
14022        \forglsentries[\@gls@type]{\@glo@label}%
14023        {%
14024          \settowidth{\dimen@}%
14025            {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14026          \ifdim\dimen@>\gls@tmplen
14027            \gls@tmplen=\dimen@
14028            \eglssetwidest{\glsentryname{\@glo@label}}%
14029          \fi
14030          \settowidth{\dimen@}%
14031            {\glsentrysymbol{\@glo@label}}%
14032          \ifdim\dimen@>#2\relax
14033            #2=\dimen@
14034          \fi
14035        }%
14036      }%
14037    }

```

**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.

```

14038    \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
14039      \dimen@=Opt\relax
14040      \gls@tmplen=Opt\relax
14041      #2=Opt\relax
14042      #3=Opt\relax
14043      \foralllglossaries[#1]{\@gls@type}%
14044      {%
14045        \forglsentries[\@gls@type]{\@glo@label}%
14046        {%
14047          \ifglsused{\@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 }%
14068 }%
14069 }

```

**eSymbolLocation** Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

14070 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
14071     \dimen@=0pt\relax
14072     \gls@tmplen=0pt\relax
14073     #2=0pt\relax
14074     #3=0pt\relax
14075     \forallglossaries[#1]{\@gls@type}%
14076     {%
14077         \forglsentries[\@gls@type]{\@glo@label}%
14078         {%
14079             \settowidth{\dimen@}%
14080                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
14081             \ifdim\dimen@>\gls@tmplen
14082                 \gls@tmplen=\dimen@
14083                 \eglssetwidest{\glsentryname{\@glo@label}}%
14084             \fi
14085             \settowidth{\dimen@}%
14086                 {\glsentrysymbol{\@glo@label}}%
14087             \ifdim\dimen@>#2\relax
14088                 #2=\dimen@
14089             \fi
14090             \settowidth{\dimen@}%
14091                 {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
14092             \ifdim\dimen@>#3\relax
14093                 #3=\dimen@
14094             \fi
14095         }%
14096     }%
14097 }

```

**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.

```

14098 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
14099     \dimen@=0pt\relax
14100     \gls@tmplen=0pt\relax
14101     #2=0pt\relax
14102     \forallglossaries[#1]{\@gls@type}%
14103     {%
14104         \forglsentries[\@gls@type]{\@glo@label}%
14105         {%

```



```

14106      \ifglsused{\@glo@label}%
14107      {%
14108      \settowidth{\dimen@}%
14109      {\glstreenamfmt{\glstentryname{\@glo@label}}}%
14110      \ifdim\dimen@>\gls@tmplen
14111      \gls@tmplen=\dimen@
14112      \eglssetwidest{\glstentryname{\@glo@label}}%
14113      \fi
14114      \settowidth{\dimen@}%
14115      {\GlsXtrFormatLocationList{\glstentrynumberlist{\@glo@label}}}%
14116      \ifdim\dimen@>#2\relax
14117      #2=\dimen@
14118      \fi
14119      }%
14120      {}%
14121      }%
14122      }%
14123      }

```

**AnyNameLocation** Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

14124 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
14125   \dimen@=0pt\relax
14126   \gls@tmplen=0pt\relax
14127   #2=0pt\relax
14128   \forallglossaries[#1]{\@gls@type}%
14129   {%
14130     \forglsentries[\@gls@type]{\@glo@label}%
14131     {%
14132       \settowidth{\dimen@}%
14133       {\glstreenamfmt{\glstentryname{\@glo@label}}}%
14134       \ifdim\dimen@>\gls@tmplen
14135       \gls@tmplen=\dimen@
14136       \eglssetwidest{\glstentryname{\@glo@label}}%
14137       \fi
14138       \settowidth{\dimen@}%
14139       {\GlsXtrFormatLocationList{\glstentrynumberlist{\@glo@label}}}%
14140       \ifdim\dimen@>#2\relax
14141       #2=\dimen@
14142       \fi
14143     }%
14144   }%
14145   }

```

**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`.

```

14146 \newcommand*{\glstxtrComputeTreeIndent}[1]{%
14147   \glstreeindent=\glstxtrtreetopindent\relax
14148   }

```

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.

```
14149 \newcommand*{\glxstrComputeTreeSubIndent}[3]{%
14150 \ifcsundef{@glswidestname\romannumeral#1}%
14151 {%
14152 \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
14153 }%
14154 {%
14155 \settowidth{#3}{\glstreenamefmt{
14156 \csname @glswidestname\romannumeral#1\endcsname\space}}%
14157 }%
14158 }
```

eeSetHangIndent Set \hangindent for top-level entries:

```
14159 \newcommand*{\glxstrAltTreeSetHangIndent}{\hangindent\glstreeindent}
```

etSubHangIndent Set \hangindent for sub-entries:

```
14160 \newcommand*{\glxstrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
14161 \renewglossarystyle{alttree}{%
14162 \renewenvironment{theglossary}%
14163 {%
14164 \glxstralttreeInit
14165 \def\@gls@prevlevel{-1}%
14166 \mbox{}\par}%
14167 {\par}%
14168 \renewcommand*{\glossaryheader}{}%
14169 \renewcommand*{\glsgroupheading}[1]{}%
14170 \renewcommand{\glossentry}[2]{%
14171 \ifnum\@gls@prevlevel=0\relax
14172 \else
14173 \glxstrComputeTreeIndent{##1}%
14174 \fi
14175 \parindent\glstreeindent
14176 \glxstrAltTreeSetHangIndent
14177 \makebox[Opt][r]%
14178 {%
14179 \glstreenamebox{\glstreeindent}%
14180 {%
14181 \glsentryitem{##1}%
14182 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
14183 }%
14184 }%
```

```

14185     \glxtralttreeSymbolDescLocation{##1}{##2}%
14186     \def\@gls@prevlevel{0}%
14187 }
14188 \renewcommand{\subglossentry}[3]{%
14189     \ifnum##1=1\relax
14190         \glssubentryitem{##2}%
14191     \fi
14192     \ifnum\@gls@prevlevel=##1\relax
14193     \else
14194         \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
14195         \ifnum\@gls@prevlevel<##1\relax
14196             \setlength\glstreeindent\gls@tmplen
14197             \addtolength\glstreeindent\parindent
14198             \parindent\glstreeindent
14199         \else
14200             \ifnum\@gls@prevlevel=0\relax
14201                 \glxtrComputeTreeIndent{##2}%
14202             \else
14203                 \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
14204             \fi
14205             \addtolength\parindent{-\glstreeindent}%
14206             \setlength\glstreeindent\parindent
14207         \fi
14208     \fi
14209     \glxtrAltTreeSetSubHangIndent{##1}%
14210     \makebox[Opt][r]{\glstreenamibox{\gls@tmplen}{%
14211         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
14212     \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
14213     \def\@gls@prevlevel{##1}%
14214 }%
14215 \renewcommand*\@glsgroupskip{\ifglsnogroupskip\else\indexspace\fi}%
14216 }
14217 }%
14218 {%
14219 }

```

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.

```

14220 \ifdef{\@glsstyle@alttreegroup}
14221 {%
14222     \renewglossarystyle{alttreegroup}{%
14223         \setglossarystyle{alttree}%
14224         \renewcommand{\glsgroupheading}[1]{\par
14225             \def\@gls@prevlevel{-1}%
14226             \hangindent0pt\relax
14227             \parindent0pt\relax
14228             \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14229         \nopagebreak\indexspace\nopagebreak
14230     }%
14231 }%

```

```

14232 }%
14233 {%
14234 }

```

Similarly for `alttreehypergroup`.

```

14235 \ifdef{\@glsstyle@alttreehypergroup}
14236 {%
14237   \renewglossarystyle{alttreehypergroup}{%
14238     \setglossarystyle{alttree}%
14239     \renewcommand*{\glossaryheader}{%
14240       \par
14241       \def\@gls@prevlevel{-1}%
14242       \hangindent0pt\relax
14243       \parindent0pt\relax
14244       \glstreenavigationfmt{\glsnavigation}\par\indexspace
14245     }%
14246     \renewcommand*{\glsgroupheading}[1]{%
14247       \par
14248       \def\@gls@prevlevel{-1}%
14249       \hangindent0pt\relax
14250       \parindent0pt\relax
14251       \glstreegroupheaderfmt
14252       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
14253       \nopagebreak\indexspace\nopagebreak
14254     }%
14255   }
14256 }%
14257 {%
14258 }

```

## 2.9 Multicolumn Styles

Adjust `mcolindexgroup` to discourage page breaks after the group headings.

```

14259 \ifdef{\@glsstyle@mcolindexgroup}
14260 {%
14261   \renewglossarystyle{mcolindexgroup}{%
14262     \setglossarystyle{mcolindex}%
14263     \renewcommand*{\glsgroupheading}[1]{%
14264       \item\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14265       \nopagebreak\indexspace\nobreak\@afterheading
14266     }%
14267   }
14268 }%
14269 {%
14270 }

```

Similarly for `mcolindexhypergroup`.

```

14271 \ifdef{\@glsstyle@mcolindexhypergroup}
14272 {%

```

```

14273 \renewglossarystyle{mcolindexhypergroup}{%
14274   \setglossarystyle{mcolindex}%
14275   \renewcommand*{\glossaryheader}{%
14276     \item\glstreenavigationfmt{\glsnavigation}%
14277     \indexspace
14278   }%
14279   \renewcommand*{\glsgroupheading}[1]{%
14280     \item\glstreegroupheaderfmt
14281       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14282     \nopagebreak\indexspace\nobreak\@afterheading
14283   }%
14284 }
14285 }%
14286 {%
14287 }

```

Similarly for mcolindexspannav.

```

14288 \ifdef{\@glsstyle@mcolindexspannav}
14289 {%
14290   \renewglossarystyle{mcolindexspannav}{%
14291     \setglossarystyle{index}%
14292     \renewenvironment{theglossary}%
14293     {%
14294       \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}}%
14295       \setlength{\parindent}{0pt}%
14296       \setlength{\parskip}{0pt plus 0.3pt}%
14297       \let\item\glstreeitem}%
14298     {\end{multicols}}}%
14299   \renewcommand*{\glsgroupheading}[1]{%
14300     \item\glstreegroupheaderfmt
14301       {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14302     \nopagebreak\indexspace\nobreak\@afterheading
14303   }%
14304 }
14305 }%
14306 {%
14307 }

```

Similarly for mcoltreegroup.

```

14308 \ifdef{\@glsstyle@mcoltreegroup}
14309 {%
14310   \renewglossarystyle{mcoltreegroup}{%
14311     \setglossarystyle{mcoltree}%
14312     \renewcommand{\glsgroupheading}[1]{\par
14313       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14314     \nopagebreak\indexspace\nobreak\@afterheading
14315   }%
14316 }
14317 }%
14318 {%

```

14319 }

Similarly for mcoltreehypergroup.

```
14320 \ifdef{\@glsstyle@mcoltreehypergroup}
14321 {%
14322   \renewglossarystyle{mcoltreehypergroup}{%
14323     \setglossarystyle{mcoltree}%
14324     \renewcommand*\{glossaryheader\}{%
14325       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
14326     }%
14327     \renewcommand*\{glsgroupheading}[1]{%
14328       \par\noindent
14329       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14330       \nopagebreak\indexspace\nobreak\@afterheading
14331     }%
14332   }
14333 }%
14334 {%
14335 }
```

Similarly for mcoltreespannav.

```
14336 \ifdef{\@glsstyle@mcoltreespannav}
14337 {%
14338   \renewglossarystyle{mcoltreespannav}{%
14339     \setglossarystyle{tree}%
14340     \renewenvironment{theglossary}%
14341     {%
14342       \begin{multicols}{\glsmcols}%
14343       [\noindent\glstreenavigationfmt{\glsnavigation}]%
14344       \setlength{\parindent}{0pt}%
14345       \setlength{\parskip}{0pt plus 0.3pt}%
14346     }%
14347     {\end{multicols}}%
14348     \renewcommand*\{glsgroupheading}[1]{%
14349       \par\noindent
14350       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14351       \nopagebreak\indexspace\nobreak\@afterheading
14352     }%
14353   }
14354 }%
14355 {%
14356 }
```

Similarly for mcoltreenonamegroup.

```
14357 \ifdef{\@glsstyle@mcoltreenonamegroup}
14358 {%
14359   \renewglossarystyle{mcoltreenonamegroup}{%
14360     \setglossarystyle{mcoltreenoname}%
14361     \renewcommand*\{glsgroupheading}[1]{\par
14362       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
14363     \nopagebreak\indexspace\nobreak\@afterheading
```

```

14364 }%
14365 }
14366 }%
14367 {%
14368 }

```

Similarly for mcoltreenonamehypergroup.

```

14369 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
14370 {%
14371   \renewglossarystyle{mcoltreenonamehypergroup}{%
14372     \setglossarystyle{mcoltreenoname}%
14373     \renewcommand*{\glossaryheader}{%
14374       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
14375     \renewcommand*{\glsgroupheading}[1]{%
14376       \par\noindent
14377       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14378       \nopagebreak\indexspace\nobreak\@afterheading}%
14379   }
14380 }%
14381 {%
14382 }

```

Similarly for mcoltreenonamespannav.

```

14383 \ifdef{\@glsstyle@mcoltreenonamespannav}
14384 {%
14385   \renewglossarystyle{mcoltreenonamespannav}{%
14386     \setglossarystyle{treenoname}%
14387     \renewenvironment{theglossary}%
14388     {%
14389       \begin{multicols}{\glsmcols}%
14390       [\noindent\glstreenavigationfmt{\glsnavigation}]]%
14391       \setlength{\parindent}{0pt}%
14392       \setlength{\parskip}{0pt plus 0.3pt}%
14393     }%
14394     {\end{multicols}}}%
14395   \renewcommand*{\glsgroupheading}[1]{%
14396     \par\noindent
14397     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14398     \nopagebreak\indexspace\nobreak\@afterheading}%
14399   }
14400 }%
14401 {%
14402 }

```

mcolalttree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{} \par which would unbalance the top of the columns.

```

14403 \ifdef{\@glsstyle@mcolalttree}
14404 {%
14405   \renewglossarystyle{mcolalttree}{%
14406     \setglossarystyle{alttree}%
14407     \renewenvironment{theglossary}%

```

```

14408   {%
14409       \glstralttreeInit
14410       \def\@gls@prevlevel{-1}%
14411       \begin{multicols}{\glsmcols}%
14412   }%
14413   {\par\end{multicols}}%
14414 }
14415 }%
14416 {%
14417 }

```

Redefine mcolalttreegroup to discourage page breaks after the group headings.

```

14418 \ifdef{\@glsstyle@mcolalttreegroup}
14419 {%
14420   \renewglossarystyle{mcolalttreegroup}{%
14421       \setglossarystyle{mcolalttree}%
14422       \renewcommand{\glsgroupheading}[1]{\par
14423           \def\@gls@prevlevel{-1}%
14424           \hangindent0pt\relax
14425           \parindent0pt\relax
14426           \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
14427           \nopagebreak\indexspace\nopagebreak
14428       }%
14429   }
14430 }%
14431 {%
14432 }

```

Similarly for mcolalttreehypergroup.

```

14433 \ifdef{\@glsstyle@mcolalttreehypergroup}
14434 {%
14435   \renewglossarystyle{mcolalttreehypergroup}{%
14436       \setglossarystyle{mcolalttree}%
14437       \renewcommand*\{\glossaryheader}{%
14438           \par
14439           \def\@gls@prevlevel{-1}%
14440           \hangindent0pt\relax
14441           \parindent0pt\relax
14442           \glstreenavigationfmt{\glsnavigation}%
14443           \par\indexspace
14444       }%
14445       \renewcommand*\{\glsgroupheading}[1]{%
14446           \par
14447           \def\@gls@prevlevel{-1}%
14448           \hangindent0pt\relax
14449           \parindent0pt\relax
14450           \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
14451           \nopagebreak\indexspace\nopagebreak
14452       }%
14453   }

```



```

14454 }%
14455 {%
14456 }

```

Similarly for mcolalttreespannav.

```

14457 \ifdef{\@glsstyle@mcolalttreespannav}
14458 {%
14459   \renewglossarystyle{mcolalttreespannav}{%
14460     \setglossarystyle{almtree}%
14461     \renewenvironment{theglossary}%
14462     {%
14463       \glsextralmtreeInit
14464       \def\@gls@prevlevel{-1}%
14465       \begin{multicols}{\@gls@col}%
14466         [\noindent\glstreenavigationfmt{\@gls@navigation}]%
14467     }%
14468     {\par\end{multicols}}%
14469     \renewcommand*\@gls@groupheading[1]{%
14470       \par
14471       \def\@gls@prevlevel{-1}%
14472       \hangindent0pt\relax
14473       \parindent0pt\relax
14474       \glstreegroupheaderfmt{\@gls@navhypertarget{##1}{\@gls@getgrouptitle{##1}}}%
14475       \nopagebreak\indexspace\nopagebreak
14476     }%
14477   }
14478 }%
14479 {%
14480 }

```

Reset the default style

```

14481 \ifx\@glossary@default@style\relax
14482 \else
14483   \setglossarystyle{\@gls@current@style}
14484 \fi

```

# 3 bookindex style (glossary-bookindex.sty)

## 3.1 Package Initialisation and Options

```
14485 \NeedsTeXFormat{LaTeX2e}
14486 \ProvidesPackage{glossary-bookindex}[2018/07/29 v1.34 (NLCT)]
```

Load required packages.

```
14487 \RequirePackage{multicol}
14488 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
14489 \newcommand{\glstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
14490 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

`bookindexsubname` Format used for sub entries.

```
14491 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

`glstrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
14492 \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.

```
14493 \newcommand*{\glstrbookindexprelocation}[1]{%
14494   \glstrifhasfield{location}{#1}%
14495   {,\glstrprelocation}%
14496   {\glstrprelocation}%
14497 }
```

`glstrsubprelocation` Separator used before location list for sub-entries.

```
14498 \newcommand*{\glstrbookindexsubprelocation}[1]{%
14499   \glstrbookindexprelocation{#1}%
14500 }
```

`glstrparentchildsep` Separator used between top-level parent and child entry.

```
14501 \newcommand{\glstrbookindexparentchildsep}{\nopagebreak}
```

`glstrparentsubchildsep` Separator used between sub-level parent and child entry.

```
14502 \newcommand{\glstrbookindexparentsubchildsep}{\glstrbookindexparentchildsep}
```

**bookindexbetween** Between two top-level entries identified by the labels in the arguments.

```
14503 \newcommand{\glxstrbookindexbetween}[2]{}
```

**indexsubbetween** Between two level 1 entries identified by the labels in the arguments.

```
14504 \newcommand{\glxstrbookindexsubbetween}[2]{}
```

**exsubsubbetween** Between two level 2 entries identified by the labels in the arguments.

```
14505 \newcommand{\glxstrbookindexsubsubbetween}[2]{}
```

**indexatendgroup** At the end of a letter group. The argument is the index of the last top-level entry.

```
14506 \newcommand{\glxstrbookindexatendgroup}[1]{}
```

**exsubatendgroup** At the end of a letter group. The argument is the index of the last level 1 entry.

```
14507 \newcommand{\glxstrbookindexsubatendgroup}[1]{}
```

**subsubatendgroup** At the end of a letter group. The argument is the index of the last level 2 entry.

```
14508 \newcommand{\glxstrbookindexsubsubatendgroup}[1]{}
```

**kindexgroupskip** Group separator.

```
14509 \newcommand{\glxstrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}
```

Format group title.

**indexformatheader** Group separator.

```
14510 \newcommand*{\glxstrbookindexformatheader}[1]{%
14511 \par{\centering\glstreegroupheaderfmt{#1}\par}%
14512 }
```

**bookindexbookmark** Book mark group heading if supported.

```
14513 \ifdef\pdfbookmark
14514 {%
14515 \newcommand*{\glxstrbookindexbookmark}[2]{%
14516 \ifdefstring{\@@glossarysec}{chapter}%
14517 {\pdfbookmark[1]{#1}{#2}}%
14518 {\pdfbookmark[2]{#1}{#2}}%
14519 }
14520 }
14521 {%
14522 \newcommand*{\glxstrbookindexbookmark}[2]{%
14523 }
```

**kindexcolspread**

```
14524 \newcommand*{\glxstrbookindexcolspread}{}%
```

**indexmulticolenv**

```
14525 \newcommand*{\glxstrbookindexmulticolenv}{multicols}
```

Define the style.

```
14526 \newglossarystyle{bookindex}{%
14527   \setglossarystyle{index}%
14528   \renewenvironment{theglossary}%
14529   {%
14530     \ifdefempty{glxstrbookindexcolspread}
14531     {%
14532       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
14533       {\glxstrbookindexcols}%
14534     }%
14535     {%
14536       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
14537       {\glxstrbookindexcols}[\glxstrbookindexcolspread]%
14538     }%
14539     \setlength{\parindent}{0pt}%
14540     \setlength{\parskip}{0pt plus 0.3pt}%
14541     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
14542     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14543     \let\@glxstr@bookindex@between\@gobble
14544     \let\@glxstr@bookindex@subbetween\@gobble
14545     \let\@glxstr@bookindex@subsubbetween\@gobble
14546     \let\@glxstr@bookindex@atendgroup\relax
14547     \let\@glxstr@bookindex@subatendgroup\relax
14548     \let\@glxstr@bookindex@subsubatendgroup\relax
14549     \let\@glxstr@bookindex@groupskip\relax
14550   }%
14551   {%
```

Do end group hooks.

```
14552     \@glxstr@bookindex@subsubatendgroup
14553     \@glxstr@bookindex@subatendgroup
14554     \@glxstr@bookindex@atendgroup
```

End multicol environment.

```
14555     \expandafter\end\expandafter{\glxstrbookindexmulticolseenv}%
14556   }%
```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```
14557   \renewcommand*{\glossaryheader}{\raggedright}%
```

Top level entry format.

```
14558   \renewcommand*{\glossentry}[2]{%
```

Do separator.

```
14559     \@glxstr@bookindex@between{##1}%
```

Update separators.

```
14560     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
14561     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14562     \let\@glxstr@bookindex@subbetween\@gobble
14563     \let\@glxstr@bookindex@subsubbetween\@gobble
14564     \edef\@glxstr@bookindex@between{%
```

```

14565      \noexpand\glxstrbookindexbetween{##1}%
14566  }%
14567  \edef\@glxstr@bookindex@atendgroup{%
14568      \noexpand\glxstrbookindexatendgroup{##1}%
14569  }%
14570  \let\@glxstr@bookindex@subatendgroup\relax
14571  \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

14572  \glstreeitem
14573      \glstryitem{##1}%
14574      \glstarget{##1}{\glxstrbookindexname{##1}}%
14575  \glxstrbookindexprelocation{##1}##2%
14576  }%
14577  \renewcommand{\subglossentry}[3]{%
14578      \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```

14579      \glstreeitem
14580      \or

```

Level 1.

```

14581      \@glxstr@bookindex@sep
14582      \@glxstr@bookindex@subbetween{##2}%
14583      \let\@glxstr@bookindex@sep\relax

```

Update separators.

```

14584      \let\@glxstr@bookindex@subsubbetween\@gobble
14585      \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14586      \edef\@glxstr@bookindex@subbetween{%
14587          \noexpand\glxstrbookindexsubbetween{##2}%
14588      }%
14589      \edef\@glxstr@bookindex@atsubendgroup{%
14590          \noexpand\glxstrbookindexatsubendgroup{##1}%
14591      }%

```

Start sub-item.

```

14592      \glstreesubitem
14593      \glssubentryitem{##2}%
14594      \else

```

All other levels.

```

14595      \@glxstr@bookindex@subsep
14596      \@glxstr@bookindex@subsubbetween{##2}%

```

Update separators.

```

14597      \let\@glxstr@bookindex@subsep\relax
14598      \edef\@glxstr@bookindex@subsubbetween{%
14599          \noexpand\glxstrbookindexsubsubbetween{##2}%
14600      }%
14601      \edef\@glxstr@bookindex@atsubsubendgroup{%
14602          \noexpand\glxstrbookindexatsubsubendgroup{##1}%
14603      }%

```

Start sub-sub-item.

```
14604 \glstreesubsubitem
14605 \fi
```

Format entry.

```
14606 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
14607 \glxtrbookindexsubprelocation{##2}##3%
14608 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
14609 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
14610 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
14611 \@glxtr@bookindex@subsubatendgroup
14612 \@glxtr@bookindex@subatendgroup
14613 \@glxtr@bookindex@atendgroup
14614 \@glxtr@bookindexgroupskip
```

Update separators.

```
14615 \let\@glxtr@bookindexgroupskip\glxtrbookindexgroupskip
14616 \let\@glxtr@bookindex@between\@gobble
14617 \let\@glxtr@bookindex@atendgroup\relax
14618 \let\@glxtr@bookindex@subatendgroup\relax
14619 \let\@glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
14620 \glxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
14621 \glxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
14622 \glxtrbookindexformatheader{\thisgrptitle}%
14623 \nopagebreak\indexspace\nopagebreak\@afterheading
14624 }%
14625 }
```

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.

```
14626 \newcommand{\glxtrbookindexthepage}{%
14627 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
14628 }
```

bookindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
14629 \newcommand*\glstrbookindexmarkentry}[1]{%
14630   \protected@write\@auxout
14631   {\let\glstrbookindexthepage\relax}%
14632   {\string\glstr@setbookindexmark{\glstrbookindexthepage}{#1}}%
14633 }
```

etbookindexmark

```
14634 \newcommand*\glstr@setbookindexmark}[2]{%
14635   \ifcsundef\glstr@idxfirstmark@#1}%
14636   {\csgdef\glstr@idxfirstmark@#1}{#2}}%
14637   {}%
14638   \csgdef\glstr@idxlastmark@#1}{#2}%
14639 }
```

indexfirstmarkfmt

```
14640 \newcommand*\glstrbookindexfirstmarkfmt}[1]{%
14641   \glstryname{#1}%
14642 }
```

indexfirstmark

```
14643 \newcommand*\glstrbookindexfirstmark}{%
14644   \letcs\glstr@label\glstr@idxfirstmark@\glstrbookindexthepage}%
14645   \ifdef\glstr@label
14646   {\glstrbookindexfirstmarkfmt\glstr@label}}%
14647   {}%
14648 }
```

indexlastmarkfmt

```
14649 \newcommand*\glstrbookindexlastmarkfmt}[1]{%
14650   \glstryname{#1}%
14651 }
```

okindexlastmark

```
14652 \newcommand*\glstrbookindexlastmark}{%
14653   \letcs\glstr@label\glstr@idxlastmark@\glstrbookindexthepage}%
14654   \ifdef\glstr@label
14655   {\glstrbookindexlastmarkfmt\glstr@label}}%
14656   {}%
14657 }
```

# Glossary

**First use** The first time a glossary entry is used (from the start of the document or after a reset) with one of the following commands: `\gls`, `\Gls`, `\GLS`, `\glspl`, `\Glspl`, `\GLSpl` or `\glsdisp`. *see* **First use flag** & **First use text**

**First use flag** A conditional that determines whether or not the entry has been used according to the rules of **first use**.

**First use text** The text that is displayed on **first use**, which is governed by the first and first-plural keys of `\newglossaryentry`. (May be overridden by `\glsdisp`.)

`makeindex` An indexing application.

`xindy` An flexible indexing application with multilingual support written in Perl.



## Change History

0.1 (2015-11-22)		\@Glsymbol@: added redefinition	74
General: Initial experimental release	5	\@Glsymbolplural@: added	
0.2 (2015-11-30)		redefinition	74
\Glsfmtshort: new	324	\@Glstext@: added redefinition	70
\glsfmtshort: new	323	\@Glsuseri@: added redefinition	75
\Glsfmtshortpl: new	324	\@Glsuserii@: added redefinition	75
\glsfmtshortpl: new	323	\@Glsuseriii@: added redefinition	75
short: switched inline full form to short		\@Glsuseriv@: added redefinition	75
(long)	226	\@Glsuseriv@: added redefinition	76
0.3 (2015-12-02)		\@Glsuservi@: added redefinition	76
\@ACRlong: added redefinition	79	\@acrlong: added redefinition	79
\@ACRlongpl: added redefinition	80	\@acrlongpl: added redefinition	80
\@ACRshort: added redefinition	77	\@acrshort: added redefinition	76
\@ACRshortpl: added redefinition	78	\@acrshortpl: added redefinition	77
\@Acrlong: added redefinition	79	\@gls@field@link: added optional	
\@Acrlongpl: added redefinition	80	argument	62
\@Acrshort: added redefinition	77	\@glsdescplural@: added redefinition	73
\@Acrshortpl: added redefinition	78	\@glsfirst@: added redefinition	70
\@GLSdesc@: added redefinition	73	\@glsfirstplural@: added redefinition	71
\@GLSdescplural@: added redefinition	73	\@glsplural@: added redefinition	71
\@GLSfirst@: added redefinition	70	\@glssymbolplural@: added	
\@GLSfirstplural@: added redefinition	72	redefinition	74
\@GLSname@: added redefinition	72	\@glsxtr@defaultnoglossarywarning:	
\@GLSplural@: added redefinition	71	new	132
\@GLSSymbol@: added redefinition	74	\@glsxtr@field@linkdefs: new	69
\@GLSSymbolplural@: added		\@glsxtr@insertdots: new	194
redefinition	74	\@print@glossary: added redefinition	129
\@GLStext@: added redefinition	69	\glsabbrvdefaultfont: renamed from	
\@GLSuseri@: added redefinition	75	\abbrvdefaultfont	200
\@GLSuserii@: added redefinition	75	\glsaccessdesc: new	158
\@GLSuseriii@: added redefinition	75	\glsaccessdescplural: new	158
\@GLSuseriv@: added redefinition	76	\glsaccessfirst: new	155
\@GLSuserv@: added redefinition	76	\glsaccessfirstplural: new	156
\@GLSuservi@: added redefinition	76	\Glsaccesslong: new	160
\@Glsdesc@: added redefinition	73	\glsaccesslong: new	160
\@Glsdescplural@: added redefinition	73	\glsaccessname: new	154
\@Glsfirst@: added redefinition	70	\glsaccessplural: new	155
\@Glsfirstplural@: added redefinition	72	\Glsaccessshort: new	159
\@Glsname@: added redefinition	72	\glsaccessshort: new	159
\@Glsplural@: added redefinition	71	\Glsaccessshortpl: new	160

\glsaccesssshortpl: new .....	160	\cGLSpl: new .....	106
\glsaccessssymbol: new .....	157	\cGLSpl@: new .....	106
\glsaccessssymbolplural: new .....	157	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new .....	154	new .....	101
\glstentryfmt: added check for short ..	61	\cGLS: new .....	105
\glslongpltok: new .....	194	\cGLSformat: new .....	106
\glsshortpltok: new .....	194	\cGLSpl: new .....	106
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new .....	106
name in \setkeys .....	196	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new .....	16
for plural .....	191	\glsenableentrycount: new .....	101
\GLSxtrlongpl: new .....	210	\glsfirstabbrvdefaultfont: new ..	200
\Glsxtrlongpl: new .....	210	\glsfirstlongdefaultfont: new ...	200
\glsxtrlongpl: new .....	209	\Glsfmtfirst: new .....	326
\glsxtrNoGlossaryWarning: new ....	21	\glsfmtfirst: new .....	326
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new .....	327
new .....	190	\glsfmtfirstpl: new .....	326
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new .....	326
new .....	190	\glsfmtplural: new .....	325
\glsxtrpostlinkendsentence: new ..	190	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new .....	209	\Glsxtrtitleshort .....	324
\Glsxtrshortpl: new .....	208	renamed from \glstentryfmtshort ..	324
\glsxtrshortpl: new .....	207	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort .....	323
\glslabeltok .....	221	renamed from \glstentryfmtshort ..	323
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok .....	219	\Glsxtrtitleshortpl .....	324
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl .....	324
redefinition of \acronymtype .....	17	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl .....	323
\Glsxtrshort .....	324	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl .....	323
\glsxtrshort .....	323	\Glsfmttext: new .....	325
\Glsfmtshortpl: changed to use		\glsfmttext: new .....	325
\glsxtrshortpl .....	324	\glshasattribute: new .....	169
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	168
\glsxtrshortpl .....	323	\glsxtremsuffix: new .....	262
\glsxtrifemptyglossary: new .....	28	\GlsXtrEnableEntryCounting: new ..	100
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new .....	103
argument .....	172	\glsxtrscfont: new .....	234
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new .....	234
argument .....	172	\glsxtrsmfont: new .....	248
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new .....	249
default type to \acronymtype ....	114	short-em: new .....	270
\newterm: fixed name argument .....	171	short-em-desc: new .....	271
0.5 (2015-12-07)		short-em-footnote: new .....	280
\cGLS: new .....	105	short-em-long: new .....	266
\cGLS@: new .....	106	short-em-long-desc: new .....	267

short-em-postfootnote: new .....	282	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new .....	245	attribute .....	314
short-sc-postfootnote: new .....	246	\Glsxtrheadtext: now uses headuc	
short-sm: new .....	252	attribute .....	317
short-sm-desc: new .....	253	\glxtrheadtext: now uses headuc	
short-sm-footnote: new .....	259	attribute .....	316
short-sm-long: new .....	250	short-em-footnote: switch off regular	
short-sm-long-desc: new .....	251	attribute if set .....	281
short-sm-postfootnote: new .....	260	short-long: switch off regular attribute	
long-noshort-em: new .....	273	if set .....	220
long-noshort-em-desc: new .....	277	short-long-desc: switch off regular	
long-noshort-sm: new .....	255	attribute if set .....	221
long-noshort-sm-desc: new .....	257	short-sc-footnote: switch off regular	
long-short-em: new .....	263	attribute if set .....	245
long-short-em-desc: new .....	264	short-sm-footnote: switch off regular	
long-short-sm: new .....	249	attribute if set .....	259
long-short-sm-desc: new .....	250	long-short: switch off regular attribute	
0.5.1 (2015-12-02)		if set .....	218
\Glsaccessstext: new .....	155	long-short-desc: switch off regular	
0.5.1 (2015-12-07)		attribute if set .....	219
\@gls@setup@default@short@access:		long-short-sc-desc: switch off regular	
removed \ifglxtruseuchead ...	314	attribute if set .....	236
\@glxtr@doaccsupp: new .....	21	footnote: switch off regular attribute if	
\Glsaccessdesc: new .....	158	set .....	222
\Glsaccessdescplural: new .....	159	postfootnote: switch off regular	
\Glsaccessfirst: new .....	156	attribute if set .....	224
\Glsaccessfirstplural: new .....	156	0.5.2 (2015-12-08)	
\Glsaccessname: new .....	154	\@GLSdesc@: added accessibility support	73
\Glsaccessplural: new .....	155	\@GLSdescplural@: added accessibility	
\Glsaccesssymbol: new .....	157	support .....	73
\Glsaccesssymbolplural: new .....	157	\@GLSfirst@: added accessibility	
\Glsxtrheadfirst: now uses headuc		support .....	70
attribute .....	318	\@GLSfirstplural@: added accessibility	
\glxtrheadfirst: now uses headuc		support .....	72
attribute .....	318	\@GLSname@: added accessibility support	72
\Glsxtrheadfirstplural: now uses		\@GLSplural@: added accessibility	
headuc attribute .....	319	support .....	71
\glxtrheadfirstplural: now uses		\@GLSsymbol@: added accessibility	
headuc attribute .....	319	support .....	74
\Glsxtrheadplural: now uses headuc		\@GLSsymbolplural@: added	
attribute .....	318	accessibility support .....	74
\glxtrheadplural: now uses headuc		\@GLStext@: added accessibility support	69
attribute .....	317	\@GLSdesc@: added accessibility support	73
\Glsxtrheadshort: now uses headuc		\@GLSdescplural@: added accessibility	
attribute .....	315	support .....	73
\glxtrheadshort: now uses headuc		\@GLSfirst@: added accessibility	
attribute .....	314	support .....	70
\Glsxtrheadshortpl: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute .....	315	support .....	72

\@Glsname@: add accessibility support ..	72	\glsdohypertarget: fixed typo in glossaries-accsupp and tidied up code to use just one \@ifpackageloaded	154
\@Glsplural@: added accessibility support .....	71	\glsentryfmt: moved \glssetabbrvfmt from \glsxtrabbrvfmt to here .....	61
\@Glsymbol@: added accessibility support .....	74	\GlsXtrEnableInitialTagging: new	186
\@Glsymbolplural@: added accessibility support .....	74	\glsxtrfieldtitlecase: new .....	173
\@Glstext@: added accessibility support	70	\GlsXtrFormatLocationList: new ...	59
\@gls@setup@default@short@access: removed \glsxtrabbrvfmt .....	211	\glsxtrnewabbrevpresetkeyhook: new .....	198
\@glsdesc@: added accessibility support	73	\glsxtrtagfont: new .....	188
\@glsdescplural@: added accessibility support .....	73	\KV@printgloss@nonumberlist: added	61
\@glsfirst@: added accessibility support .....	70	\mfu@checkword@do: added .....	187
\@glsfirstplural@: added accessibility support .....	71	\setabbreviationstyle: added check for post-definition style switch ....	214
\@Glsname@: added accessibility support	72	0.5.3 (2015-12-09)	
\@Glsplural@: added accessibility support .....	71	\@glsxtr@autoindex@at: new .....	184
\@Glsymbol@: added accessibility support .....	74	\@glsxtr@autoindex@encap: new ...	184
\@Glsymbolplural@: added accessibility support .....	74	\@glsxtr@autoindex@esc: new .....	184
\@Glstext@: added accessibility support	69	\@glsxtr@autoindex@level: new ...	184
\@glsxtr@activate@initialtagging: new .....	188	\@glsxtr@autoindex@setname: new .	182
\@glsxtr@do@titlecaps@warn: new .	188	\@glsxtr@doabbreviationsdef: new .	17
\@glsxtr@tag: new .....	188	\glsdescwidth: added .....	58
\glossaryentrynumbers: added .....	59	\glsdohypertarget: removed \GlsXtrNoGlsWarningNoAutoMakeMain .....	131
\Glossentrydesc: added .....	186	\glsPagelistwidth: added .....	58
\Glossentryname: added .....	177	\glsxtrdoautoindexname: new .....	182
\Glossentrysymbol: added .....	186	\glsxtrpostnamehook: new .....	179
\glossentrysymbol: added .....	186	\if@glsxtr@format@override: new .	181
\GLSaccessdesc: new .....	158, 166	\ProvidesGlossariesExtraLang: new	330
\GLSaccessdescplural: new ...	159, 166	\RequireGlossariesExtraLang: new	329
\GLSaccessfirst: new .....	156, 165	0.5.4 (2015-12-15)	
\GLSaccessfirstplural: new ..	156, 165	\@newglossaryentry@defunitcounters: new .....	107
\GLSaccesslong: new .....	160, 167	\@GLSxtr@p@acrlong@: new .....	92
\GLSaccesslongpl: new .....	161, 167	\@GLSxtr@p@acrlongpl@: new .....	92
\Glsaccesslongpl: new .....	161	\@GLSxtr@p@acrshort@: new .....	92
\glsaccesslongpl: new .....	161	\@GLSxtr@p@acrshortpl@: new .....	92
\GLSaccessname: new .....	154, 164	\@GLSxtr@p@long@: new .....	91
\GLSaccessplural: new .....	155, 165	\@GLSxtr@p@longpl@: new .....	92
\GLSaccessshort: new .....	159, 166	\@GLSxtr@p@plural@: new .....	90
\GLSaccessshortpl: new .....	160, 167	\@GLSxtr@p@short@: new .....	91
\GLSaccesssymbol: new .....	157, 165	\@GLSxtr@p@shortpl@: new .....	91
\GLSaccesssymbolplural: new .	158, 166	\@GLSxtr@p@text@: new .....	90
\GLSaccessstext: new .....	155, 164	\@GlsXtrEnableOnTheFly: new .....	54
		\@Glsxtr: new .....	55
		\@Glsxtr@p@acrlong@: new .....	92

\@Glsxtr@p@acrlongpl@: new .....	92	\glstdonohyperlink: added .....	89
\@Glsxtr@p@acrshort@: new .....	92	\glsenableentryunitcount: new ...	109
\@Glsxtr@p@acrshortpl@: new .....	92	\glshasattribute: added check for	
\@Glsxtr@p@long@: new .....	91	entry's existence .....	169
\@Glsxtr@p@longpl@: new .....	91	\glusifattribute: added check for	
\@Glsxtr@p@plural@: new .....	90	entry's existence .....	169
\@Glsxtr@p@short@: new .....	90	\glspostlinkhook: added existence	
\@Glsxtr@p@shortpl@: new .....	91	check .....	189
\@Glsxtr@p@text@: new .....	90	\Glsxtr: new .....	55
\@Glsxtrpl: new .....	56	\glxtr: new .....	55
\@alt@glshyp@opt: new .....	86	\glxtrcat: new .....	55
\@glscalt@hyp@opt: new .....	86	\glxtrdowrglossaryhook: new .....	86
\@glscalt@hyp@opt@char: new .....	86	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new .....	86	new .....	112
\@glsc@increment@currunitcount:		\GlsXtrEnableOnTheFly: new .....	54
new .....	108	\Glsxtrpl: new .....	56
\@glsc@local@increment@currunitcount:		\glxtrpl: new .....	55
new .....	108	\glxtrpostlocalreset: new .....	100
\@glsc@setdefault@glslink@opts:		\glxtrpostlocalunset: new .....	99
new .....	83	\glxtrpostreset: new .....	100
\@glsxtr: new .....	55	\glxtrpostunset: new .....	98
\@glsxtr@addunitcounter: new ....	107	\glxtrprotectlinks: new .....	89
\@glsxtr@currunitcount: new ....	109	\GlsXtrSetAltModifier: new .....	86
\@glsxtr@ifunitcounter: new ....	107	\GlsXtrSetDefaultGlsOpts: new ....	85
\@glsxtr@p@acrlong@: new .....	92	\glxtrstarflywarn: new .....	54
\@glsxtr@p@acrlongpl@: new .....	92	\GlsXtrWarning: new .....	56
\@glsxtr@p@acrshort@: new .....	92	\MakeAcronymsAbbreviations: now	
\@glsxtr@p@acrshortpl@: new .....	92	disables \setacronymstyle .....	114
\@glsxtr@p@long@: new .....	91	1.0 (2016-01-24)	
\@glsxtr@p@longpl@: new .....	91	\@glsxtr@autoindexcrossrefs: new .	16
\@glsxtr@p@plural@: new .....	90	\@glsxtr@idx@displaynumberlist:	
\@glsxtr@p@short@: new .....	90	new .....	123
\@glsxtr@p@shortpl@: new .....	91	\@glsxtr@idx@entrynumberlist: new	124
\@glsxtr@p@text@: new .....	90	\@glsxtr@noidx@displaynumberlist:	
\@glsxtr@prevunitcount: new ....	109	new .....	123
\@glsxtr@setentryunitcountunsetattr:		\@glsxtr@noidx@entrynumberlist:	
new .....	112	new .....	124
\@glsxtr@unitcountlist: new ....	107	\@glsxtr@noidx@numberlistloop:	
\@glsxtrpl: new .....	55	new .....	123
\@newglossaryentryposthook: added		\@glsxtr@reg@glosslist: new .....	115
empty see value if not set and added		\makeglossaries: new .....	116
'see' to field key map .....	45	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new ....	54	\glxtrdiscardperiod: added check	
\cGlsformat: added .....	106	for first use .....	191
\cGlsformat: added .....	106	short-desc: fixed typo in	
\cGlsplformat: added .....	107	\glxtrinlinfullformat and	
\cGlsplformat: added .....	106	added missing second argument ...	228
\glstdisablehyper: added .....	88	1.02 (2016-04-25)	
\glstdohyperlink: added .....	87	\@glsxtr@current@style: new .....	57



\@glsfirst@: set abbreviation and regular format .....	70	\glsxtrregularfont: new .....	61
\@glsfirstplural@: set abbreviation and regular format .....	71	\glsxtruserfield: new .....	284
\@glsname@: set abbreviation and regular format .....	72	\glsxtruserparen: new .....	284
\@glsplural@: set abbreviation and regular format .....	71	\glsxtrusersuffix: new .....	285
\@glsymbol@: set regular format .....	74	\GlsXtrWarnDeprecatedAbbrStyle:	
\@glsymbolplural@: set regular format .....	74	new .....	216
\@glstext@: set abbreviation and regular format .....	69	short-em-long-em: new .....	268
\@glsxtr@deprecated@abbrstyle:		short-em-long-em-desc: new .....	269
new .....	216	short-em-nolong: new .....	271
\@glsxtr@do@style: new .....	22	short-em-nolong-desc: new .....	272
\@glsxtr@doloctag: new .....	61	short-em-postfootnote: renamed from “postfootnote-em” .....	282
\@glsxtr@idx@entrynumberlist:		short-footnote: new .....	224
switched from \let to \newcommand .....	124	short-long-user: new .....	291
\@glsxtr@pagetag: new .....	60	short-long-user-desc: new .....	293
\@glsxtr@pagetag: new .....	60	short-nolong: new .....	227
\@glsxtr@preloctag: new .....	61	short-nolong-desc: new .....	229
\@glsxtr@postloctag: new .....	60	short-postfootnote: new .....	226
\@glsxtr@preloctag: new .....	60	short-sc-footnote: renamed from “footnote-sc” .....	245
\glossentrydesc: added glossdescfont attribute check .....	173	short-sc-nolong: new .....	239
\Glossentryname: added glossnamefont attribute check .....	177	short-sc-nolong-desc: new .....	241
\glossentryname: added glossnamefont attribute check .....	175	short-sc-postfootnote: renamed from “postfootnote-sc” .....	246
moved post name hook inside condition .....	177	short-sm-footnote: renamed from “footnote-sm” .....	259
\glsabbrvmfont: new .....	262	short-sm-nolong: new .....	253
\glsabbrvuserfont: new .....	284	short-sm-nolong-desc: new .....	255
\glsfirstabbrvmfont: new .....	262	short-sm-postfootnote: renamed from “postfootnote-sm” .....	260
\glsfirstabbrvuserfont: new .....	284	\letabbreviationstyle: new .....	216
\glsfirstlongemfont: new .....	263	\newabbreviationstyle: bug fix: corrected test for existence .....	215
\glsfirstlonguserfont: new .....	285	long-em-noshort-em: new .....	275
\glsifnotregularcategory: new ...	170	long-em-noshort-em-desc: new ....	278
\glslongdefaultfont: new .....	200	long-em-short-em: new .....	264
\glslongemfont: new .....	263	long-em-short-em-desc: new .....	265
\glslongfont: new .....	200	long-noshort: new .....	233
\glslonguserfont: new .....	285	long-noshort-desc: new .....	232
\glsxtrassignfieldfont: new .....	69	long-noshort-em: renamed from “long-em” .....	273
\GlsXtrEnablePreLocationTag: new .	59	long-noshort-em-desc: renamed from “long-desc-em” .....	277
\glsxtrfirstscfont: new .....	234	long-noshort-sc: renamed from “long-sc” .....	241
\glsxtrfirstsmfont: new .....	248	long-noshort-sc-desc: renamed from “long-desc-sc” .....	243
\glsxtrlongshortdescsort: new ...	219	long-noshort-sm: renamed from “long-sm” .....	255
\glsxtrpostnamehook: added category check .....	179		

long-noshort-sm-desc: renamed from		General: disabled docdef key at the start	
\long-desc-sm .....	257	of the document .....	27
long-short-user: new .....	285	docdef option changed to choice .....	14
long-short-user-desc: new .....	291	\glstr@usesee: new .....	46
\renewabbreviationstyle: new ....	216	\glstrusesee: new .....	45
style: new .....	22	\glstruseseeformat: new .....	46
1.05 (2016-06-10)		\if@glstrdocdefrestricted: new ..	15
\eglssetwidest: new .....	385	1.07 (2016-08-15)	
\glFindWidestAnyName: new .....	387	@@glstrp: new .....	93
\glFindWidestAnyNameLocation:		\GLSfirst@: added check for	
new .....	393	nohyperfirst attribute .....	71
\glFindWidestAnyNameSymbol: new	390	\GLSfirstplural@: added check for	
\glFindWidestAnyNameSymbolLocation:		nohyperfirst attribute .....	72
new .....	392	\GLSxtrp: new .....	94
\glFindWidestLevelTwo: new ....	389	\@Glsfirst@: added check for	
\glFindWidestUsedAnyName: new ..	387	nohyperfirst attribute .....	70
\glFindWidestUsedAnyNameLocation:		\@Glsfirstplural@: added check for	
new .....	392	nohyperfirst attribute .....	72
\glFindWidestUsedAnyNameSymbol:		\@Glsxtrp: new .....	93
new .....	390	\@glspreglossaryhook: added	
\glFindWidestUsedAnyNameSymbolLocation:		\glossxtrsetpopts .....	189
new .....	391	\@glfirst@: added check for	
\glFindWidestUsedLevelTwo: new .	388	nohyperfirst attribute .....	70
\glFindWidestUsedTopLevelName:		\@glfirstplural@: added check for	
new .....	386	nohyperfirst attribute .....	71
\glfirstlongfootnotefont: new ..	222	\@glxtrinmark: new .....	312
\glsgetwidestname: new .....	386	\@glxtrnotinmark: new .....	312
\glsgetwidestsubname: new .....	386	\@glxtrp: new .....	93
\glslongfootnotefont: new .....	222	\@glxtrp@opt: new .....	92
\glxtrAltTreeIndent: new .....	384	\glossxtrsetpopts: new .....	93
\glxtralttreeInit: new .....	385	\glsp: new .....	95
\glxtrAltTreePar: new .....	384	\glsp: new .....	95
\glxtrAltTreeSetHangIndent: new	394	\glxtr@entry@p: new .....	94
\glxtrAltTreeSetSubHangIndent:		\glxtrabbrvfootnote: new .....	222
new .....	394	\glxtrchecknohyperfirst: new ....	70
\glxtralttreeSubSymbolDescLocation:		\glxtrfieldtitlecasecs: new ....	173
new .....	384	\glxtrifinmark: new .....	311
\glxtralttreeSymbolDescLocation:		\GLSxtrp: new .....	96
new .....	384	\Glsxtrp: new .....	95
\glxtrComputeTreeIndent: new ...	393	\glxtrp: new .....	94
\glxtrComputeTreeSubIndent: new	394	\glxtrsetpopts: new .....	93
\glxtrtreetopindent: new .....	385	short-long-desc: added text key ....	221
short-em-long: fixed incorrect font used		fixed misspelling of \glabbrvfont in	
by long form .....	267	plural key .....	221
\xglsetwidest: new .....	385	long-short-desc: added missing text	
1.06 (2016-06-18)		key .....	219
\@glsoifexistsorwarn: new .....	15	fixed misspelling of \glabbrvfont .	219
\@glxtr@docdefval: new .....	15	footnote: changed first forms to use	
\@glxtr@usesee: new .....	46	\glfirstlongfootnotefont ...	222



postfootnote: removed \footnote		\@printglossary: redefined to save	
from first keys .....	224	options .....	121
switched from \glsfirstlongfont to		\glsxtr@makeglossaries: new .....	122
\glsfirstlongfootnotefont ...	225	1.10 (2016-12-17)	
\RestoreAcronyms: modified		\@GLSp1@: fixed bug caused by typo in	
\@gls@link@checkfirsthyper to		command name .....	63
set \glsxtrifwasfirstuse .....	115	1.11 (2017-01-19)	
1.08 (2016-12-13)		\@glsxtr@do@redef@forglsentries:	
\@glsxtr@record: new .....	8	new .....	6
\@GLS@: added \@glsxtr@record .....	63	\@glsxtr@noidx@do: new .....	143
\@GLSp1@: added \@glsxtr@record ...	63	\@glsxtr@redef@forglsentries: new .	6
\@Gls@: added \@glsxtr@record .....	63	\@glsxtr@shortcutsval: new .....	20
\@GLSp1@: added \@glsxtr@record ...	63	\@glsxtr@unsrt@getgrouptitle: new	142
\@gls@: added \@glsxtr@record .....	62	\@print@noidx@glossary: added	
\@gls@link@: added		redefinition .....	126
\@glsxtr@record .....	64	\glsxtr@addloclistfield: added	
\@gls@field@link: added		group key .....	13
\@glsxtr@record .....	62	added location key .....	12
\@gls@saveentrycounter: new .....	27	\glsxtr@fields: new .....	135
\@glsdisp: added \@glsxtr@record ..	63	\glsxtr@linkprefix: new .....	135
\@GLSp1@: added \@glsxtr@record ...	62	\glsxtr@org@newignoredglossary:	
\@glsxtr@dorecord: new .....	10	new .....	40
\@glsxtr@err@undefaction: new .....	6	\glsxtr@s@newignoredglossary: new	41
\@glsxtr@record: new .....	7	\glsxtr@shortcutsval: new .....	135
\@glsxtr@warn@onexistsordo: new ...	6	\glsxtr@texencoding: new .....	135
\@glsxtr@warn@undefaction: new ....	6	\glsxtr@writefields: new .....	135
\@print@unsrt@glossary: new .....	140	\GlsXtrLoadResources: new .....	134
record: added record package option ...	13	\glsxtrpageref: new .....	38
\glsadd: added \@glsxtr@record ....	68	\glsxtrresourcefile: changed	
\glsdoifexists: now defines		extension to .gls tex .....	134
\glslabel .....	44	\newignoredglossary: added starred	
\glsxtr@do@wrglossary: new .....	27	version .....	40
\glsxtr@addloclistfield: new .....	12	1.12 (2017-02-03)	
\glsxtr@indexonly@saveentrycounter:		\@glsxtr@recordcounter: new .....	11
new .....	12	\@gls@preglossaryhook: check for	
\glsxtr@record: new .....	137	definition .....	188
\glsxtr@resource: new .....	135	\@glsxtr@counterrecordhook: new .	137
\glsxtr@saveentrycounter: new ....	27	\@glsxtr@display@loc: new .....	127
\glsxtr@setup@record: new .....	12	\@glsxtr@docounterrecord: new ...	137
\glsxtrassignfieldfont: added check		\@glsxtr@longnewglossaryentry:	
for existence .....	69	new .....	40
\glsxtrresourcefile: new .....	134	\@glsxtr@noop@recordcounter: new .	11
\printunsrtglossaries: new .....	139	\@glsxtr@op@recordcounter: new ...	11
\printunsrtglossary: new .....	139	\@glsxtr@provide@storagekey: new .	28
1.09 (2016-12-16)		\@glsxtr@s@longnewglossaryentry:	
\@glsxtr@gettype: new .....	122	new .....	39
\@glsxtr@mixed@assign@sortkey:		\@glsxtrentryfmt: new .....	31
new .....	122	\@glsxtrindexaliased: new .....	84
		\@glsxtrsetaliasnoindex: new .....	84

\@newglossaryentryposthook: added check for alias key .....	49	\glxtrindexaliased: new .....	85
\@no@glxtrindexaliased: new .....	84	\GlsXtrLetField: new .....	34
\@printunsrtglossary: new .....	139	\GlsXtrLetFieldToField: new .....	34
General: added target key to printgloss family .....	121	\GlsXtrLoadResources: removed restriction on only one per document	134
\apptoglossarypreamble: new .....	38	\glxtrlocrangefmt: new .....	128
\csGlsXtrLetField: new .....	34	\glxtrpostlongdescription: new ..	40
\eGlsXtrSetField: new .....	35	\glxtrprovidestoragekey: new ....	28
\gGlsXtrSetField: new .....	35	\GlsXtrRecordCounter: new .....	137
\glsdohyperlink: added check for alias field .....	88	\glxtrresourcecount: new .....	134
\glsnoidxdisplayloc: added redefinition .....	127	\glxtrresourcefile: added catcode change for @ .....	134
\glissettoctitle: added patch .....	41	\glxtrsetaliasnoindex: new .....	84
\glxtr@counterrecord: new .....	137	\GlsXtrSetField: new .....	34
\glxtr@langtag: new .....	135	\glxtrsetfieldifexists: new ....	34
\glxtr@newabbreviation: new ....	195	\glxtrunsrtdo: new .....	142
\glxtr@org@newignoredglossary: Added check for existence .....	40	\glxtrusefield: new .....	34
\glxtr@pluralsuffixes: new ....	135	\glxtrusefield: new .....	34
\glxtr@provideignoredglossary: new .....	42	short-postlong-user: new .....	288
\glxtr@s@newignoredglossary: Added check for existence .....	41	short-postlong-user-desc: new ...	290
\glxtr@s@provideignoredglossary: new .....	43	\longnewglossaryentry: added starred version .....	39
\glxtrabbrvpluralsuffix: new ...	200	long-postshort-user: new .....	286
\glxtralias: new .....	49	long-postshort-user-desc: new ...	288
\glxtrcopytoglossary: new .....	43	postdot: new .....	16
\glxtrdeffield: new .....	34	\pretoglossarypreamble: new .....	38
\glxtrdisplayendloc: new .....	127	\print@noop@unsrtglossaryunit: new .....	142
\glxtrdisplayendlochook: new ...	128	\print@op@unsrtglossaryunit: new	142
\glxtrdisplaysingleloc: new ....	127	\printunsrtglossary: added starred form .....	139
\glxtrdisplaystartloc: new ....	127	\printunsrtglossaryhandler: new .	141
\glxtrredefield: new .....	34	\printunsrtglossaryunit: new ....	12
\glxtrentryfmt: new .....	30	\printunsrtglossaryunitsetup: new	142
\glxtrfieldddolistloop: new .....	31	\provideignoredglossary: new ....	42
\glxtrfieldforlistloop: new ....	32	\s@glxtr@provide@storagekey: new	29
\glxtrfielddininlist: new .....	32	\s@printunsrtglossary: new .....	139
\glxtrfieldlistadd: new .....	31	\xGlsXtrSetField: new .....	35
\glxtrfieldlistadd: new .....	31	1.13 (2017-02-07)	
\glxtrfieldlistgadd: new .....	31	\@glsdisp: removed	
\glxtrfieldlistxadd: new .....	31	\@glxtr@org@glsdisp .....	63
\glxtrfieldxifinlist: new .....	32	\glxtrsetaliasnoindex: switched to \providecommand .....	84
\glxtrfmt: new .....	29	1.14 (2017-04-18)	
\GlsXtrFmtDefaultOptions: new ....	29	\@gls@link: added redefinition .....	65
\GlsXtrFmtField: new .....	29	\@gls@noidx@getgrouptitle: new ..	124
\glxtrifkeydefined: new .....	28	\@gls@removespaces: new .....	128
		\@glxtr@do@automake@err: new ...	136
		\@glxtr@org@gloautosee: new ....	25

\@glxtr@record: added third arg .....	7	postfootnote: fixed spelling of	
\@glxtr@recordsee: new .....	11	\glsabbrvfont .....	224
General: added \glsadd option		1.16 (2017-06-15)	
theHvalue .....	68	\@glo@autosee: added redefinition ....	26
added \glsadd option thevalue ....	68	\@gls@noidx@getgrouptitle: fixed	
\glsdisablehyper: added redefinition .	88	bug .....	124
\glsenableentrycount: fixed		\@glxtr@addunusedxrefs: added	
assignment of \@cGls@ .....	102	check for seealso field .....	50
\glsenableentryunitcount: fixed		\@glxtr@checkgroup: use \csuse	
assignment of \@cGls@ .....	110	instead of \csname .....	143
\glsnavigation: new .....	126	\@glxtr@dorecordnodefer: new ....	11
\glxtr@org@getgrouptitle: new ..	125	\@print@unsrt@glossary: corrected	
\glxtr@recordsee: new .....	7	misspelt command .....	140
\glxtr@writefields: added check for		\@printunsrt@glossary@handler:	
automake .....	136	new .....	141
\glxtrdisplayendloc: added check		record: added check for	
for empty format .....	127	\@gls@setupsort@none .....	14
\glxtrgetgrouptitle: new .....	125	\gls@checkseeallowed: added	
\glxtrinitwrgloss: new .....	64	redefinition .....	26
\glxtrlocationhyperlink: new ...	128	\glxtr@writefields: added	
\glxtrsetgrouptitle: new .....	125	\providecommand lines .....	135
\glxtrsupphypernumber: new ....	129	\glxtrautoindex: new .....	182
\ifglxtrwrglossbefore: new .....	64	\glxtrautoindexassignsort: new .	182
1.15 (2017-05-10)		\glxtrautoindexentry: new .....	182
\@glxtr@dorecord: corrected		\glxtrindexseealso: new .....	47
premature expansion of \@glslocref	10	\glxtrseealsolabels: new .....	49
short-em-long-em: fixed spelling of		\glxtrseelist: new .....	47
\glsabbrvfont .....	268	\glxtruseseealso: new .....	46
short-long: fixed spelling of		\glxtruseseealsoformat: new ....	46
\glsabbrvfont .....	220	\seealso: new .....	47
short-long-user: fixed spelling of		autoseeindex: new .....	16
\glsabbrvfont .....	292	1.17 (2017-08-09)	
short-postlong-user: fixed spelling of		\@gls@setup@default@short@access:	
\glsabbrvfont .....	289	removed some inconsistencies in the	
short-postlong-user-desc: fixed		abbreviation styles .....	217
spelling of \glsabbrvfont .....	290	\@glxtr@mark@wordseps: new ....	195
long-em-short-em: fixed spelling of		\@glxtr@markwordseps: new .....	195
\glsabbrvfont .....	265	\@glxtr@noidx@displaynumberlist:	
long-postshort-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont .....	286	\glxtrundeftag .....	123
long-postshort-user-desc: fixed		\@glxtr@noidx@entrynumberlist:	
spelling of \glsabbrvfont .....	288	replace hard-coded ?? with	
long-short: fixed spelling of		\glxtrundeftag .....	124
\glsabbrvfont .....	218	\@glxtr@noidx@numberlistloop:	
long-short-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont .....	285	\glxtrundeftag .....	124
footnote: fixed spelling of		\@glxtrifhyphenstart: new .....	294
\glsabbrvfont .....	222	\glsabbrvhyphenfont: new .....	294
		\glsabbrvonlyfont: new .....	307

<code>\glsabbrvscfont:new</code> .....	234	<code>short-hyphen-postlong-hyphen-desc:</code>	
<code>\glsabbrvsmfont:new</code> .....	248	<code>new</code> .....	307
<code>\glsabbrvuserfont:initialised to</code>		<code>short-long-user-desc:corrected first</code>	
<code>default font</code> .....	284	<code>forms</code> .....	293
<code>\glsfirstabbrvhyphenfont:new</code> ...	294	<code>short-nolong-desc-noreg:new</code> ...	229
<code>\glsfirstabbrvonlyfont:new</code> .....	307	<code>short-nolong-noreg:new</code> .....	227
<code>\glsfirstabbrvscfont:new</code> .....	234	<code>long-em-noshort-em-desc-noreg:</code>	
<code>\glsfirstabbrvsmfont:new</code> .....	248	<code>new</code> .....	280
<code>\glsfirstlonghyphenfont:new</code> ...	294	<code>long-em-noshort-em-noreg:new</code> ...	276
<code>\glsfirstlongonlyfont:new</code> .....	308	<code>long-hyphen-noshort-desc-noreg:</code>	
<code>\glslonghyphenfont:new</code> .....	294	<code>new</code> .....	296
<code>\glslongonlyfont:new</code> .....	307	<code>long-hyphen-postshort-hyphen:new</code>	300
<code>\glslonguserfont:initialised to default</code>		<code>long-hyphen-postshort-hyphen-desc:</code>	
<code>font</code> .....	285	<code>new</code> .....	301
<code>\glxtr@newabbreviation:added</code>		<code>long-hyphen-short-hyphen:new</code> ...	295
<code>\glxtrorgshort and</code>		<code>long-hyphen-short-hyphen-desc:</code>	
<code>\glxtrorglong</code> .....	196	<code>new</code> .....	295
<code>\GlsXtrDefineAcShortcuts:new</code> ...	18	<code>long-noshort-desc-noreg:new</code> ...	232
<code>\glxtrgenabbrvfmt:added check for</code>		<code>long-noshort-noreg:new</code> .....	233
<code>\ifglxtrininsertinside</code> .....	211	<code>long-only-short-only:new</code> .....	308
<code>\glxtrhyphensuffix:new</code> .....	294	<code>long-only-short-only-desc:new</code> ..	309
<code>\glxtrifhyphenstart:new</code> .....	293	<code>long-short-user-desc:corrected first</code>	
<code>\glxtrlonghyphen:new</code> .....	299	<code>forms</code> .....	291
<code>\glxtrlonghyphennoshort:new</code> ...	296	1.18 (2017-08-10)	
<code>\glxtrlonghyphenshort:new</code> .....	294	<code>stylemods:changed default value to</code>	
<code>\glxtrlongshortdescname:new</code> ...	219	<code>"default"</code> .....	22
<code>\glxtronlydescname:new</code> .....	309	1.19 (2017-09-09)	
<code>\glxtronlydescsort:new</code> .....	309	<code>\@glxtr@defaultnumberformat:new</code> .	7
<code>\glxtronlysuffix:new</code> .....	308	<code>\@glxtr@dorecord:Use</code>	
<code>\glxtrparen:new</code> .....	198	<code>\@glxtr@recordlocref instead of</code>	
<code>\glxtrposthyphenlong:new</code> .....	304	<code>\@glxtr@locref</code> .....	10
<code>\glxtrposthyphenshort:new</code> .....	299	<code>\@glxtr@dorecordnodefer:Use</code>	
<code>\glxtrposthyphensubsequent:new</code>	299	<code>\theglentrycounter for the</code>	
<code>\glxtrshortdescname:new</code> .....	228	<code>location rather than \@glxtr@locref</code> ..	11
<code>\glxtrshorthyphen:new</code> .....	304	<code>\@glxtr@record@setting:new</code> .....	13
<code>\glxtrshorthyphenlong:new</code> .....	302	<code>\@glxtr@record@setting@alsoindex:</code>	
<code>\glxtrshortlongdescname:new</code> ...	221	<code>new</code> .....	13
<code>\glxtrshortlongdescsort:new</code> ...	221	<code>\@glxtrifhasfield:new</code> .....	32
<code>\Glsxtrsubsequentfmt:new</code> .....	214	General:added <code>\glslink</code> option	
<code>\glxtrsubsequentfmt:new</code> .....	213	<code>theHvalue</code> .....	65
<code>\Glsxtrsubsequentplfmt:new</code> .....	214	added <code>\glslink</code> option <code>thevalue</code> ...	64
<code>\glxtrsubsequentplfmt:new</code> .....	213	<code>\glxtr@writefields:removed</code>	
<code>\glxtrword:new</code> .....	195	<code>double-quotes around \@jobname</code> ..	136
<code>\glxtrwordsep:new</code> .....	195	<code>\glxtrdoautoindexname:changed</code>	
<code>short-hyphen-long-hyphen:new</code> ...	303	<code>format test</code> .....	182
<code>short-hyphen-long-hyphen-desc:</code>		<code>\glxtrhyperlink:new</code> .....	88
<code>new</code> .....	303	<code>\glxtrifhasfield:new</code> .....	32
<code>short-hyphen-postlong-hyphen:new</code>	305	<code>\GlsXtrSetDefaultNumberFormat:</code>	
		<code>new</code> .....	7

\s@glxstrifhasfield: new .....	32	\@glxtrsetaliasnoindex: changed to use \glxstrifhasfield instead of \ifglshasfield .....	84
1.20 (2017-09-11)		\@glxtrwrglossmark: new .....	24
\@glxtrhypernameprefix: new ....	122	\@rGLS: new .....	150
\glsdohypertarget: added redefinition	122	\@rGLS@: new .....	150
\printunsrtglossaryunitsetup: switched from redefining \glolinkprefix to \@glxtrhypernameprefix .....	142	\@rGLSpl: new .....	151
1.21 (2017-11-03)		\@rGLSpl@: new .....	151
\@@glxtr@record: added check for default options .....	9	\@rGLs: new .....	150
\@@glxtrwrglossmark: new .....	24	\@rGLspl: new .....	150
\@gls@setup@default@short@access: modified index to remove hard coded \space .....	379	\@rGLspl@: new .....	150
modified list to remove hard coded \space .....	368	\@rgls: new .....	149
moved conditional outside of \glsgroupskip .....	371–378	\@rgls@: new .....	149
redefined altlistgroup to discourage breaks after group headings .....	369	\@rglspl: new .....	149
redefined altlisthypergroup to discourage breaks after group headings .....	370	\@rglspl@: new .....	149
redefined indexgroup to discourage breaks after group headings .....	380	General: adjusted mcolalmtree .....	399
redefined indexhypergroup to discourage breaks after group headings .....	380	new .....	402
redefined listgroup to discourage breaks after group headings .....	369	redefined almtreegroup to discourage breaks after group headings .....	395
redefined listhypergroup to discourage breaks after group headings .....	369	redefined almtreehypergroup to discourage breaks after group headings .....	396
\@glslink: changed \let to \def .....	89	redefined mcolalmtreegroup to discourage breaks after group headings .....	400
\@glxtr@checkgroup: new .....	143	redefined mcolalmtreehypergroup to discourage breaks after group headings .....	400
\@glxtr@defpostpunc: new .....	16	redefined mcolalmtreepannav to discourage breaks after group headings .....	401
\@glxtr@do@record@wrglossary: new .....	8	redefined mcolindexgroup to discourage breaks after group headings .....	396
\@glxtr@dosee@alsoindex@glossary: new .....	25	redefined mcolindexhypergroup to discourage breaks after group headings .....	396
\@glxtr@doseeglossary: new .....	25	redefined mcolindexspannav to discourage breaks after group headings .....	397
\@glxtr@noidx@do: removed code dealing with the group .....	144	redefined mcoltreegroup to discourage breaks after group headings .....	397
\@glxtr@record@setting@off: new .	13	redefined mcoltreehypergroup to discourage breaks after group headings .....	398
\@glxtr@record@setting@only: new	13	redefined mcoltreenamegroup to discourage breaks after group	
\@glxtr@rglstrigger@record: new	148		
\@glxtrglossentry: new .....	137		
\@glxtrnewgls: new .....	145		

headings .....	398	\glstrbookindexatendgroup: new ..	403
redefined		\glstrbookindexbetween: new ....	403
mcoltreenonamehypergroup to		\glstrbookindexbookmark: new ...	403
discourage breaks after group		\glstrbookindexcols: new .....	402
headings .....	399	\glstrbookindexcolspread: new ..	403
redefined mcoltreenonamespannav to		\glstrbookindexfirstmark: new ..	407
discourage breaks after group		\glstrbookindexfirstmarkfmt: new	407
headings .....	399	\glstrbookindexformatheader: new	403
redefined mcoltreepannav to		\glstrbookindexgroupskip: new ..	403
discourage breaks after group		\glstrbookindexlastmark: new ...	407
headings .....	398	\glstrbookindexlastmarkfmt: new	407
redefined treenonamegroup to		\glstrbookindexmarkentry: new ..	407
discourage breaks after group		\glstrbookindexname: new .....	402
headings .....	383	\glstrbookindexparentchildsep:	
redefined treenonamehypergroup to		new .....	402
discourage breaks after group		\glstrbookindexparentschildsep:	
headings .....	383	new .....	402
debug: new .....	24	\glstrbookindexprelocation: new	402
\gglsssetwidest: new .....	385	\glstrbookindexsubatendgroup:	
\gl:disablehyper: added check for		new .....	403
existence .....	88	\glstrbookindexsubbetween: new ..	403
changed to use \def rather than \let ..	88	\glstrbookindexsubname: new ....	402
\gl:dohypertarget: redefined		\glstrbookindexsubprelocation:	
treegroup to discourage breaks after		new .....	402
group headings .....	381	\glstrbookindexsubsubatendgroup:	
redefined treehypergroup to		new .....	403
discourage breaks after group		\glstrbookindexsubsubbetween:	
headings .....	382	new .....	403
\gl:enablehyper: changed to use \def		\glstrbookindexthepage: new ....	406
rather than \let .....	89	\glstrdetoklocation: new .....	147
\Glsfmtname: new .....	324	\glstrenablerecordcount: new ...	147
\gl:fmtname: new .....	324	\glstrglossentry: new .....	137
\glshex: new .....	331	\glstrgroupfield: new .....	143
\glslstchildpostlocation: new ..	368	\Glsxrheadname: new .....	316
\glslstchildprelocation: new ...	368	\glstrheadname: new .....	315
\glslstprelocation: new .....	368	\GlsXtrIfFieldEqStr: new .....	35
\gl:navhyperlink: patched .....	87	\glstriflabelinlist: new .....	141
\gl:seeitemformat: new .....	46	\glstrifrecordtrigger: new .....	148
\glsshowtarget: new .....	25	\glstrindexseealso: added check	
\glstreechildprelocation: new ...	379	that the entry exists .....	47
\glstreeprelocation: new .....	379	\glstrinithyperoutside: new .....	65
\glstriggerrecordformat: new ....	149	\GlsXtrLocationRecordCount: new ..	147
\gl:useabbrvfont: new .....	211	\glstrnewgls: new .....	144, 145
\gl:suselongfont: new .....	211	\glstrnewGLSlike: new .....	146
\gl:xt@do@alsoindex@wrglossary:		\glstrnewglslike: new .....	146
new .....	8	\glstrnewrgls: new .....	146
\gl:xt@org@@@do@wrglossary: new ..	27	\glstrnewrGLSlike: new .....	146
\gl:xt@org@dohyperlink: new .....	86	\glstrnewrglslike: new .....	146
\gl:xt@setbookindexmark: new ...	407	\glstrprelocation: new .....	367, 402

\GlsXtrRecordCount: new .....	146	\@glxtr@orgprintglossary: changed	
\glxtrrecordtriggervalue: new ..	147	explicit \let for \nopostdesc to	
\glxtrresourcefile: now disables		\glxtractivatenopost .....	120
record key .....	134	\@glxtrglossentryother: new ....	139
\glxtrresourceinit: new .....	134	\glossentrynameother: new .....	180
\GlsXtrSetRecordCountAttribute:		\glseeitemformat: switched check	
new .....	147	from regular to short .....	46
\glxtrtitlename: new .....	316	\glxtr@setaccessdisplay: new ...	179
\glxtrtitleorpdforheading: new .	312	\glxtr@writefields: provide	
\GlsXtrTotalRecordCount: new ....	146	\glxtr@record in aux file .....	135
\glxtrwrglossmark: new .....	24	\glxtractivatenopost: new .....	120
short-em: new .....	270	\glxtrbookindexprelocation:	
short-sc: corrected first letter		removed check for no post dot ....	402
uppercasing .....	239	\glxtrglossentryother: new ....	138
short-sm: corrected first letter		\glxtrnopostpunc: new .....	121
uppercasing .....	253	1.23 (2017-11-12)	
shortcuts: ac .....	21	\@@glxtrfmt: added check for indexing	30
\ifglxtr@hyperoutside: new .....	65	added grouping .....	30
all: new .....	366	new .....	30
nolong-short: new .....	230	\@glxtr@nopostpunc@postdesc: new	121
nolong-short-em: new .....	272	\@glxtr@restore@postpunc: new ..	121
nolong-short-noreg: new .....	230	\@glxtrentryfmt: fixed missing label	
nolong-short-sc: new .....	241	argument .....	31
nolong-short-sm: new .....	255	\@glxtrfmt: new .....	29
nopostdot: new .....	16	\eglupdatewidest: new .....	386
postpunc: new .....	16	\gglupdatewidest: new .....	385
\printunsrtglossaryentryprocesshook:		\glupdatewidest: new .....	385
new .....	141	\GlsXtrDefineAbbreviationShortcuts:	
\printunsrtglossarypredoglossary:		changed \newabbr definition to use	
new .....	141	\providecommand .....	18
\printunsrtglossaryskipentry: new	141	\GlsXtrDefineAcShortcuts: changed	
\rGLS: new .....	150	\newabbr definition to use	
\rGls: new .....	149	\providecommand .....	19
\rgls: new .....	149	\glxtrfmtdisplay: new .....	30
\rGLSformat: new .....	152	\glxtrifcustomdiscardperiod: new	190
\rGlsformat: new .....	151	\GlsXtrIfFieldUndef: new .....	33
\rglsformat: new .....	151	\glxtrrestorepostpunc: new ....	121
\rGLSpl: new .....	151	\s@glxtrfmt: new .....	30
\rGlspl: new .....	150	\s@glxtrfmt: new .....	30
\rglspl: new .....	149	\xglupdatewidest: new .....	386
\rGLSplformat: new .....	152	1.24 (2017-11-14)	
\rglsplformat: new .....	151	\glxtr@setsort .....	68
\s@glxtrifhasfield: switched from		\glxtrforcsvfield: new .....	32
\ifdef to \ifundef .....	32	\glxtrlocalsetgrouptitle: new ..	125
1.22 (2017-11-08)		1.25 (2017-11-14)	
\@glxtr@nopostpunc: new .....	121	\glxtrbookindexmulticolsest: new	403
		1.25 (2017-11-24)	
		\glxtrapostnamehook: new .....	179
		\glxtrfootnotename: new .....	222

\glxtrlongnoshortdescname: new .	231	\glxtrGeneralLatinIIrules: new .	341
\glxtrlongnoshortname: new . . . . .	233	\glxtrGeneralLatinIrules: new ..	341
\glxtrlongshortname: new . . . . .	217	\glxtrGeneralLatinIVrules: new .	343
\glxtrlongshortuserdescname: new	288	\glxtrGeneralLatinVIIIrules: new	346
\glxtronlyname: new . . . . .	308	\glxtrGeneralLatinVIIrules: new	345
\glxtrpostlinkAddDescOnFirstUse:		\glxtrGeneralLatinVrules: new .	344
changed to use \glxtrparen . . . .	190	\glxtrGeneralLatinVrules: new ..	343
\glxtrpostlinkAddSymbolOnFirstUse:		\glxtrgeneralpuncIIrules: new ..	340
changed to use \glxtrparen . . . .	190	\glxtrgeneralpuncIrules: new ...	339
\glxtrshortlongname: new . . . . .	220	\glxtrgeneralpuncrules: new . . . .	339
\glxtrshortlonguserdescname: new	290	\glxtrhyphenrules: new . . . . .	339
\glxtrshortnolongname: new . . . . .	226	\glxtrLatinA: new . . . . .	346
1.26 (2018-01-05)		\glxtrLatinAA: new . . . . .	348
\@glxtr@do@inc@linkcount: new ..	152	\glxtrLatinAELigature: new . . . .	348
\glslinkpresetkeys: new . . . . .	65	\glxtrLatinE: new . . . . .	346
\glxtr@inc@linkcount: new . . . . .	65	\glxtrLatinEszettSs: new . . . . .	348
\GlsXtrEnableLinkCounting: new ..	153	\glxtrLatinEszettSz: new . . . . .	348
\GlsXtrIfLinkCounterDef: new . . . .	153	\glxtrLatinEth: new . . . . .	348
\glxtrinclinkcounter: new . . . . .	153	\glxtrLatinH: new . . . . .	347
\GlsXtrLinkCounterName: new . . . .	153	\glxtrLatinI: new . . . . .	347
\GlsXtrLinkCounterValue: new . . . .	153	\glxtrLatinInsularG: new . . . . .	349
\GlsXtrTheLinkCounter: new . . . . .	153	\glxtrLatinK: new . . . . .	347
1.27 (2018-02-26)		\glxtrLatinL: new . . . . .	347
\@glset@setup@default@short@access:		\glxtrLatinLslash: new . . . . .	349
added glossaries-extra-bib2gls.sty .	330	\glxtrLatinM: new . . . . .	347
\@glxtrdialecthook: new . . . . .	27	\glxtrLatinN: new . . . . .	347
\Alpha: new . . . . .	333	\glxtrLatinO: new . . . . .	347
\Beta: new . . . . .	333	\glxtrLatinOELigature: new . . . .	348
\Chi: new . . . . .	334	\glxtrLatinOslash: new . . . . .	349
\Digamma: new . . . . .	334	\glxtrLatinP: new . . . . .	347
\Epsilon: new . . . . .	333	\glxtrLatinS: new . . . . .	347
\Eta: new . . . . .	333	\glxtrLatinSchwa: new . . . . .	348
\glxtr@loaddialect: new . . . . .	330	\glxtrLatinT: new . . . . .	347
\glxtrBasicDigitrules: new . . . .	363	\glxtrLatinThorn: new . . . . .	348
\glxtrcombiningdiacriticIIrules:		\glxtrLatinWynn: new . . . . .	349
new . . . . .	337	\glxtrLatinX: new . . . . .	348
\glxtrcombiningdiacriticIIrules:		\glxtrMathGreekIIrules: new . . . .	355
new . . . . .	337	\glxtrMathGreekIrules: new . . . .	354
\glxtrcombiningdiacriticIrules:		\glxtrMathItalicAlpha: new . . . .	359
new . . . . .	337	\glxtrMathItalicBeta: new . . . .	359
\glxtrcombiningdiacriticIVrules:		\glxtrMathItalicChi: new . . . . .	362
new . . . . .	338	\glxtrMathItalicDelta: new . . . .	360
\glxtrcombiningdiacriticrules:		\glxtrMathItalicEpsilon: new ...	360
new . . . . .	336	\glxtrMathItalicEta: new . . . . .	360
\glxtrcontrolrules: new . . . . .	335	\glxtrMathItalicGamma: new . . . .	360
\glxtrcurrencyrules: new . . . . .	340	\glxtrMathItalicGreekIIrules:	
\glxtrdigitrules: new . . . . .	363	new . . . . .	351
\glxtrfractionrules: new . . . . .	364	\glxtrMathItalicGreekIrules: new	350
\glxtrGeneralLatinIIIrules: new	342	\glxtrMathItalicIota: new . . . . .	360



<code>\glxtrMathItalicKappa:new</code> . . . . .	361	<code>\glxtrUpPi:new</code> . . . . .	358
<code>\glxtrMathItalicLambda:new</code> . . . . .	361	<code>\glxtrUpPsi:new</code> . . . . .	359
<code>\glxtrMathItalicLowerGreekIIrules:</code>		<code>\glxtrUpRho:new</code> . . . . .	358
<code>new</code> . . . . .	353	<code>\glxtrUpSigma:new</code> . . . . .	358
<code>\glxtrMathItalicLowerGreekIrules:</code>		<code>\glxtrUpTau:new</code> . . . . .	359
<code>new</code> . . . . .	352	<code>\glxtrUpTheta:new</code> . . . . .	357
<code>\glxtrMathItalicMu:new</code> . . . . .	361	<code>\glxtrUpUpsilon:new</code> . . . . .	359
<code>\glxtrMathItalicNabla:new</code> . . . . .	363	<code>\glxtrUpXi:new</code> . . . . .	358
<code>\glxtrMathItalicNu:new</code> . . . . .	361	<code>\glxtrUpZeta:new</code> . . . . .	357
<code>\glxtrMathItalicOmega:new</code> . . . . .	362	<code>\Iota:new</code> . . . . .	333
<code>\glxtrMathItalicOmicron:new</code> . . . . .	361	<code>\Kappa:new</code> . . . . .	333
<code>\glxtrMathItalicPartial:new</code> . . . . .	363	<code>\Mu:new</code> . . . . .	333
<code>\glxtrMathItalicPhi:new</code> . . . . .	362	<code>\Nu:new</code> . . . . .	333
<code>\glxtrMathItalicPi:new</code> . . . . .	361	<code>\Omicron:new</code> . . . . .	333
<code>\glxtrMathItalicPsi:new</code> . . . . .	362	<code>\omicron:new</code> . . . . .	334
<code>\glxtrMathItalicRho:new</code> . . . . .	361	<code>\Rho:new</code> . . . . .	333
<code>\glxtrMathItalicSigma:new</code> . . . . .	362	<code>\Tau:new</code> . . . . .	333
<code>\glxtrMathItalicTau:new</code> . . . . .	362	<code>\Upalpha:new</code> . . . . .	334
<code>\glxtrMathItalicTheta:new</code> . . . . .	360	<code>\Upbeta:new</code> . . . . .	334
<code>\glxtrMathItalicUpperGreekIIrules:</code>		<code>\Upchi:new</code> . . . . .	335
<code>new</code> . . . . .	352	<code>\Upsilon:new</code> . . . . .	334
<code>\glxtrMathItalicUpperGreekIrules:</code>		<code>\Upeta:new</code> . . . . .	334
<code>new</code> . . . . .	351	<code>\Upiota:new</code> . . . . .	334
<code>\glxtrMathItalicUpsilon:new</code> . . . . .	362	<code>\Upkappa:new</code> . . . . .	334
<code>\glxtrMathItalicXi:new</code> . . . . .	361	<code>\Upmu:new</code> . . . . .	334
<code>\glxtrMathItalicZeta:new</code> . . . . .	360	<code>\Upnu:new</code> . . . . .	334
<code>\glxtrMathUpGreekIIrules:new</code> . . . . .	350	<code>\Upomicron:new</code> . . . . .	334
<code>\glxtrMathUpGreekIrules:new</code> . . . . .	349	<code>\upomicron:new</code> . . . . .	335
<code>\glxtrnonprintablerules:new</code> . . . . .	336	<code>\Uprho:new</code> . . . . .	334
<code>\glxtrprovidecommand:new</code> . . . . .	331	<code>\Uptau:new</code> . . . . .	335
<code>\glxtrspacerules:new</code> . . . . .	336	<code>\Upzeta:new</code> . . . . .	334
<code>\glxtrSubScriptDigitrules:new</code> . . . . .	363	<code>\Zeta:new</code> . . . . .	333
<code>\glxtrSuperScriptDigitrules:new</code> . . . . .	363		
<code>\glxtrUpAlpha:new</code> . . . . .	356	1.28 (2018-03-06)	
<code>\glxtrUpBeta:new</code> . . . . .	356	<code>\@glxtr@docdefval</code> : changed from	
<code>\glxtrUpChi:new</code> . . . . .	359	count register to macro . . . . .	15
<code>\glxtrUpDelta:new</code> . . . . .	356	<code>\@glxtrdialecthook</code> : save and restore	
<code>\glxtrUpDigamma:new</code> . . . . .	357	<code>\TrackLangRequireDialectPrefix</code>	
<code>\glxtrUpEpsilon:new</code> . . . . .	357	. . . . .	364
<code>\glxtrUpEta:new</code> . . . . .	357	<code>\glxtrredeffield</code> : changed <code>\csedef</code> to	
<code>\glxtrUpGamma:new</code> . . . . .	356	<code>\protected@csedef</code> . . . . .	34
<code>\glxtrUpIota:new</code> . . . . .	357	<code>\glxtrlocalsetgrouptitle</code> : changed	
<code>\glxtrUpKappa:new</code> . . . . .	357	<code>\csedef \protected@csedef</code> . . . . .	125
<code>\glxtrUpLambda:new</code> . . . . .	358	<code>\glxtrsetgrouptitle</code> : changed	
<code>\glxtrUpMu:new</code> . . . . .	358	<code>\csxdef \protected@csxdef</code> . . . . .	125
<code>\glxtrUpNu:new</code> . . . . .	358	1.29 (2018-04-09)	
<code>\glxtrUpOmega:new</code> . . . . .	359	<code>\@glx@removespaces</code> : added expansion	128
<code>\glxtrUpOmicron:new</code> . . . . .	358	<code>\@glxtr@dorecord</code> : don't suppress	
<code>\glxtrUpPhi:new</code> . . . . .	359	expansion of <code>\@glxrecordloc</code> if	
		counter isn't page . . . . .	11

\@glxstr@wrglossary@locationhyperlink: new .....	23	\Glsxtrlongpl: added \@glxstr@record .....	210
\glxstr@inc@wrglossaryctr: new ...	23	\glxtrlongpl: added \@glxstr@record .....	209
\glxstr@wrglossarylocation: new ..	331	\GLSxtrshort: added \@glxstr@record .....	205
\GlsXtrBibTeXEntryAliases: new ..	332	\Glsxtrshort: added \@glxstr@record .....	204
\glxtrfieldforlistloop: corrected argument order in \forlistcsloop	32	\glxtrshort: added \@glxstr@record .....	204
\GlsXtrIndexCounterLink: new ....	331	\GLSxtrshortpl: added \@glxstr@record .....	209
\GlsXtrInternalLocationHyperlink: new .....	23	\Glsxtrshortpl: added \@glxstr@record .....	208
\GlsXtrProvideBibTeXFields: new ..	332	\glxtrshortpl: added \@glxstr@record .....	208
indexcounter: new .....	23	\GlsXtrStartUnsetBuffering: new ..	98
\setentrycounter: new .....	128	\GlsXtrStopUnsetBuffering: new ...	99
1.30 (2018-04-25)		indexcounter: added check for wrglossary counter .....	23
\@@glxstr@record: added check for post-key hook .....	9	\s@GlsXtrStopUnsetBuffering: new ..	99
added check for pre-key hook .....	9	1.31 (2018-05-09)	
\@GLSxtr@fullpl: added \@glxstr@record .....	203	\@GlsXtrStartUnsetBuffering: new ..	98
\@GlsXtrStopUnsetBuffering: new ..	99	\@gl@ifaccessattribute@set: new	162
\@Glsxtr@fullpl: added \@glxstr@record .....	203	\@gl@initaccesskeys: new ...	161, 167
\@glxstr@dorecord: don't suppress expansion of \@glarecordlocref ..	11	\@gl@setup@default@short@access: new .....	162, 167
\@glxstr@full: added \@glxstr@record .....	200	\@glxstr@record@noglossarywarning: new .....	134
\@glxstr@fullpl: added \@glxstr@record .....	202	\@glxstrbuffer@nodup@unset: new ..	99
\@glxstr@glossadd@postkeys: new ..	10	General: added prefix key for glslink	65
\@glxstr@glossadd@prekeys: new ...	10	added prefix key for printgloss ..	122
\@glxstr@glslink@postkeys: new ...	10	changed \let to \def .....	122
\@glxstr@glslink@prekeys: new ....	10	\glsaddeach: new .....	69
\@glxstr@local@textformat: new ...	65	\glsapturedgroup: new .....	331
\@glxstr@unset: new .....	98	\glsdefpostdesc: new .....	189
\@glxstrbuffer@unset: new .....	99	\glsdefpostlink: new .....	190
\glsadd: added \glsaddpostsetkeys	68	\glsdefpostname: new .....	179
added \glsaddpresetkeys .....	68	\glsdohypertarget: bug fix: ensure that new version is picked up .....	122
\glsaddpostsetkeys: new .....	68	\glslistdesc: new .....	368
\glsaddpresetkeys: new .....	68	\glslocalreseteach: new .....	100
\glsuserdescription: new .....	285	\glslocalunseteach: new .....	100
\glxtrabbreviationfont: new ....	62	\glstreechilddesc: new .....	381
\GlsXtrDualBackLink: new .....	332	\glstreechildsymbol: new .....	381
\GlsXtrDualField: new .....	331	\glstreedefaultnamefmt: new ....	378
\GlsXtrExpandedFmt: new .....	65	\glstreegroupheaderfmt: added redefinition .....	378
\GLSxtrlong: added \@glxstr@record	207	\glstreenamefmt: added redefinition	378
\Glsxtrlong: added \@glxstr@record	206		
\glxtrlong: added \@glxstr@record	206		
\GLSxtrlongpl: added \@glxstr@record .....	211		

\glstreenavigationfmt:added		\GlsXtrStandaloneGlossaryType:	
redefinition .....	379	new .....	138
\glstreenonamechilddesc:new ....	382	\GlsXtrStandaloneSubEntryItem:	
\glstreenonamesymbol:new .....	382	new .....	138
\glstreesymbol:new .....	381	\s@GlsXtrStartUnsetBuffering:new	98
\glsxtr@newabbreviation:added		1.32 (2018-05-24)	
\ExtraCustomAbbreviationFields		\GlsXtrForeignText:new .....	36
.....	196	\GlsXtrForeignTextField:new .....	37
\GlsXtrForUnsetBufferedList:new .	99	\GlsXtrUnknownDialectWarning:new	37
\GlsXtrIfFieldCmpNum:new .....	33	1.33 (2018-07-26)	
\GlsXtrIfFieldEqNum:new .....	33	\ifglused: added redefinition .....	39
\GlsXtrIfFieldEqXpStr:new .....	35	1.34 (2018-07-29)	
\GlsXtrIfFieldNonZero:new .....	33	\gls@begindocdefs: atom .....	51
\GlsXtrIfHasNonZeroChildCount:		\GlsXtrIfUnusedOrUndefined:new ..	27
new .....	331	\glsxtrNoGlossaryWarning: added	
\GlsXtrIfXpFieldEqXpStr:new .....	35	package warning .....	21
\glsxtrpostlinkAddSymbolDescOnFirstUse:		\if@glsxtrdocdefrestricted:	
new .....	191	changed to allow for atom as well ...	15
\GlsXtrRecordWarning:new .....	132	docdef: atom .....	15
\glsxtrRevertTocMarks:new .....	311		

# Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
\\$	331
\.	16, 17, 190, 378
\@	51, 134
\@cGLS@	102, 111
\@cGLSpl@	102, 111
\@cGls@	102, 110
\@cGlspl@	102, 111
\@cglS@	102, 110
\@cglSpl@	102, 110
\@do@wrglossary	8, 10, 117
\@do@wrglossary	12–14, 27, 68, 84
\@glo@assign@sortkey	123
\@glo@list	6
\@glo@type	140
\@glossarysec	403
\@gls@expand@field	29
\@glslocalreset	100
\@glslocalunset	99
\@glsreset	100
\@glsunset	98
\@glsxtr@autoindex@escspch	184, 185
\@glsxtr@checkspch	183, 185, 186
\@glsxtr@disabledflycommand	57
\@glsxtr@org@postdescription	121
\@glsxtr@record	14
\@glsxtr@recordcounter	13, 14, 137
\@glsxtrfmt	29, 30
\@glsxtrp	93, 94
\@glsxtrpostloctag	59
\@glsxtrpreloctag	59, 60
\@glsxtrwrglossmark	8, 9, 12, 25, 27, 47, 52, 117
\@newglossaryentry@defcounters	101
\@newglossaryentry@defunitcounters	109
\@par	384
\@ACRlong	90
\@ACRlongpl	90
\@ACRshort	90
\@ACRshortpl	90
\@Acrlong	90
\@Acrlongpl	90
\@Acrshort	90
\@Acrshortpl	90
\@GLS@	89, 105, 106, 151
\@GLSdesc@	73
\@GLSpl@	89, 105, 106, 151
\@GLSplural@	90
\@GLSSymbol@	74
\@GLSxtr@	90
\@GLSxtr@full	201
\@GLSxtr@fullpl	203
\@GLSxtr@p@acrlong@	90
\@GLSxtr@p@acrlongpl@	90
\@GLSxtr@p@acrshort@	90
\@GLSxtr@p@acrshortpl@	90
\@GLSxtr@p@long@	89
\@GLSxtr@p@longpl@	90
\@GLSxtr@p@plural@	89
\@GLSxtr@p@short@	89
\@GLSxtr@p@shortpl@	89
\@GLSxtr@p@text@	89
\@GLSxtrlong	89, 207
\@GLSxtrlongpl	90, 210, 211
\@GLSxtrp	97
\@GLSxtrshort	89, 205
\@GLSxtrshortpl	89, 209
\@Gls@	89, 104, 105, 150
\@Gls@acentryname	113
\@Gls@entry@field	81, 96, 180
\@Gls@entryname	113
\@GlsXtrEnableOnTheFly	54
\@GlsXtrStartUnsetBuffering	98
\@GlsXtrStopUnsetBuffering	99

<code>\@Glspl@</code> .....	89, 104, 105, 150	<code>\@end@glxtr@addunused</code> .....	50, 51
<code>\@Glsplural@</code> .....	90	<code>\@end@glxtr@gettype</code> .....	119, 122
<code>\@Glstext@</code> .....	90	<code>\@end@glxtr@usesee</code> .....	45, 46
<code>\@Glsxtr</code> .....	55, 56	<code>\@end@glxtrifhyphenstart</code> .....	293, 294
<code>\@Glsxtr@full</code> .....	201	<code>\@endfortrue</code> .....	32, 179, 214
<code>\@Glsxtr@fullpl</code> .....	202	<code>\@firstofone</code> .....	69, 140, 174, 181, 187
<code>\@Glsxtr@p@acrlong@</code> .....	90	<code>\@firstofthree</code> .....	63,
<code>\@Glsxtr@p@acrlongpl@</code> .....	90		69, 77–80, 86, 201, 202, 204, 206, 208, 210
<code>\@Glsxtr@p@acrshort@</code> .....	90	<code>\@firstoftwo</code> 70–74, 78, 80, 83, 86, 115, 179,	
<code>\@Glsxtr@p@acrshortpl@</code> .....	90		180, 192, 193, 201–203, 208–211, 312, 313
<code>\@Glsxtr@p@long@</code> .....	89	<code>\@for</code> .....	6, 22, 32, 51, 69, 100, 101, 113,
<code>\@Glsxtr@p@longpl@</code> .....	89		116, 119, 126, 140, 147, 153, 172, 179, 187
<code>\@Glsxtr@p@plural@</code> .....	89	<code>\@glo@alias</code> .....	48, 49
<code>\@Glsxtr@p@short@</code> .....	89	<code>\@glo@assign@sortkey</code> .....	119
<code>\@Glsxtr@p@shortpl@</code> .....	89	<code>\@glo@autosee</code> .....	25
<code>\@Glsxtr@p@text@</code> .....	89	<code>\@glo@autoseehook</code> .....	49
<code>\@Glsxtrlong</code> .....	89, 206	<code>\@glo@category</code> .....	107
<code>\@Glsxtrlongpl</code> .....	89, 210	<code>\@glo@check@sortallowed</code> .....	119
<code>\@Glsxtrp</code> .....	96	<code>\@glo@counterprefix</code> .....	10, 11, 128
<code>\@Glsxtrpl</code> .....	56	<code>\@glo@countunit</code> .....	107
<code>\@Glsxtrshort</code> .....	89, 204	<code>\@glo@default@sorttype</code> .....	119
<code>\@Glsxtrshortpl</code> .....	89, 208	<code>\@glo@desc</code> .....	40
<code>\@acrlong</code> .....	90	<code>\@glo@descplural</code> .....	40
<code>\@acrlongpl</code> .....	90	<code>\@glo@group</code> .....	13
<code>\@acrshort</code> .....	90	<code>\@glo@label</code> .....	
<code>\@acrshortpl</code> .....	90		12, 13, 28, 45, 48–50, 81, 88, 387–393
<code>\@addtoreset</code> .....	152	<code>\@glo@location</code> .....	12
<code>\@afterheading</code> .....		<code>\@glo@loclist</code> .....	12
	369, 370, 380, 382–384, 396–399, 406	<code>\@glo@name</code> .....	182
<code>\@alt@glshyp@opt</code> .....	86	<code>\@glo@no@assign@sortkey</code> .....	122
<code>\@auxout</code> .....	11, 12, 52, 60, 61,	<code>\@glo@parent</code> .....	388, 389
	103, 111, 116, 117, 129, 130, 134–137, 407	<code>\@glo@see</code> .....	45, 46, 49–51
<code>\@bibgl@restoreat</code> .....	134	<code>\@glo@seealso</code> .....	48, 49
<code>\@cGLS</code> .....	105	<code>\@glo@sort</code> .....	182
<code>\@cGLS@</code> .....	102, 105, 111	<code>\@glo@sorttype</code> .....	119, 126
<code>\@cGLSpl</code> .....	106	<code>\@glo@text</code> .....	63
<code>\@cGLSpl@</code> .....	102, 106, 111	<code>\@glo@thislettergrp</code> .....	143
<code>\@cGls@</code> .....	102, 110	<code>\@glo@thisvalue</code> .....	284
<code>\@cGlspl@</code> .....	102, 111	<code>\@glo@tmp</code> .....	28, 47, 81
<code>\@cgls@</code> .....	102, 110	<code>\@glo@type</code> .....	50, 87, 113, 116, 119,
<code>\@cglspl@</code> .....	102, 110		120, 122, 126, 127, 129, 130, 132, 133, 140
<code>\@disable@onlypremakeg</code> .....	117	<code>\@glo@types</code> .....	170, 171, 386–393
<code>\@do@auxoutstuff</code> .....	129, 130	<code>\@glossary@default@style</code> .	57, 58, 120, 401
<code>\@do@gl@getcounterprefix</code> .....	10, 11	<code>\@glossarystyle</code> .....	120
<code>\@do@gl@see</code> .....	49, 50	<code>\@gls@</code> .....	89, 104, 105, 149
<code>\@do@newglossaryentry</code> .....	113, 114, 198	<code>\@gls@@link</code> .....	64
<code>\@do@seeglossary</code> .....	13, 14, 25, 52, 117	<code>\@gls@ReturnAfterFi</code> .....	128
<code>\@do@wrglossary</code> .....	67, 148	<code>\@gls@actualchar</code> .....	183
<code>\@empty</code> ....	69, 77–80, 121, 128, 183, 201–211	<code>\@gls@adjustmode</code> .....	68

<code>\@gls@alt@hyp@opt</code> .....	86	<code>\@gls@long</code> .....	196
<code>\@gls@alt@hyp@opt@char</code> .....	86	<code>\@gls@longpl</code> .....	194, 196, 197
<code>\@gls@alt@hyp@opt@keys</code> .....	86	<code>\@gls@map</code> .....	179
<code>\@gls@automake</code> .....	119	<code>\@gls@nameaccess</code> .....	161, 163
<code>\@gls@between</code> .....	126	<code>\@gls@nohyperlist</code> .....	41–43
<code>\@gls@checkedmkidx</code> .....	183, 185	<code>\@gls@noidx@do</code> .....	126
<code>\@gls@checkmkidxchars</code> .....	47, 182	<code>\@gls@noidx@getgrouptitle</code> .....	140
<code>\@gls@codepage</code> .....	130	<code>\@gls@noidx@nosanitizesort</code> .....	118
<code>\@gls@counter</code> .....	9, 11, 23, 66, 68, 84, 148	<code>\@gls@noidx@sanitizesort</code> .....	118
<code>\@gls@currentlettergroup</code> ...	126, 140, 143	<code>\@gls@noidx@loclist@finalsep</code> .....	123
<code>\@gls@declareoption</code> .....	5	<code>\@gls@noidx@loclist@prev</code> .....	123
<code>\@gls@default@longpl</code> .....	196, 197	<code>\@gls@noidx@loclist@sep</code> .....	123
<code>\@gls@deffile</code> .....	52	<code>\@gls@noref@warn</code> .....	117, 127
<code>\@gls@doautomake</code> .....	119, 136	<code>\@gls@org@glsnoidx@displayloc</code> ..	123, 124
<code>\@gls@doautomake@err</code> .....	136	<code>\@gls@org@glsseeformat</code> .....	123, 124
<code>\@gls@enablesavenonumberlist</code> .....	51	<code>\@gls@preglossaryhook</code> .....	120, 187
<code>\@gls@encapchar</code> .....	183	<code>\@gls@prevlevel</code> .....	394–396, 400, 401
<code>\@gls@entry@count</code> .....	102, 103	<code>\@gls@quotechar</code> .....	183
<code>\@gls@entry@field</code> .....		<code>\@gls@reference</code> .....	52, 53, 116, 117
.....	29, 34, 49, 81, 94–97, 102, 138, 139	<code>\@gls@restoreat</code> .....	51
<code>\@gls@entry@unitcount</code> .....	111	<code>\@gls@saveentrycounter</code> ..	13, 14, 27, 66, 68, 148
<code>\@gls@field@font</code> .....	69–76	<code>\@gls@see@noindex</code> .....	26, 134
<code>\@gls@field@link</code> .....	69–76, 81, 82	<code>\@gls@setdefault@glslink@opts</code> .....	
<code>\@gls@firstaccess</code> .....	161, 164	.....	9, 30, 66, 85
<code>\@gls@getcounterprefix</code> .....	10, 11	<code>\@gls@setsort</code> .....	66, 68
<code>\@gls@getgrouptitle</code> .....	125, 140	<code>\@gls@setupsort@none</code> .....	14
<code>\@gls@grptitle</code> .....	87, 126	<code>\@gls@short</code> .....	196, 197
<code>\@gls@hyp@opt</code> .....		<code>\@gls@shortaccess</code> .....	161–164
.....	81, 82, 86, 105, 106, 145, 149–151, 200–210	<code>\@gls@shortaccesspl</code> .....	161–163
<code>\@gls@hyp@opt@cs</code> .....	86	<code>\@gls@shortpl</code> .....	194, 197
<code>\@gls@ifaccessattribute@set</code> .....	162	<code>\@gls@sort</code> .....	143
<code>\@gls@ifinlist</code> .....	142	<code>\@gls@textaccess</code> .....	161, 163
<code>\@gls@increment@currcount</code> .....	102	<code>\@gls@thislabel</code> .....	69, 100
<code>\@gls@increment@currunitcount</code> .....	110	<code>\@gls@thisval</code> .....	179
<code>\@gls@initaccesskeys</code> .....	196	<code>\@gls@tmp</code> .....	35, 126
<code>\@gls@keymap</code> ..	12, 13, 28, 45, 48, 81, 135, 179	<code>\@gls@tmpb</code> .....	185
<code>\@gls@label</code> ..	8, 9, 11, 52, 85, 86, 117, 137, 214	<code>\@gls@type</code> .....	117–119, 214, 387–393
<code>\@gls@levelchar</code> .....	183	<code>\@gls@write@entrycounts</code> .....	102
<code>\@gls@link</code> .....	30, 62–64, 77–81, 201–211	<code>\@gls@write@entryunitcounts</code> .....	111
<code>\@gls@link@checkfirsthyper</code> .....	63, 115	<code>\@gls@write@entryunitcounts@do</code> ....	112
<code>\@gls@link@label</code> .....	66, 148	<code>\@gls@writedef</code> .....	52
<code>\@gls@link@nocheckfirsthyper</code> .....		<code>\@gls@xref</code> .....	12, 47
.....	62, 77–80, 200–211	<code>\@gls@abbrv@current@abbreviation</code> ..	196, 211
<code>\@gls@link@opts</code> .....	66	<code>\@gls@sacronymlists</code> .....	113
<code>\@gls@list</code> .....	126	<code>\@gls@doifexistsorwarn</code> ...	15, 175–178, 180
<code>\@gls@local@increment@currcount</code> ...	102	<code>\@gls@entry</code> .....	52, 103, 111, 112
<code>\@gls@local@increment@currunitcount</code> ..	110	<code>\@gls@link</code> .....	67, 87–89
<code>\@gls@location</code> .....	143, 144	<code>\@gls@localreset</code> .....	100
<code>\@gls@loclist</code> .....	123, 124, 143, 144	<code>\@gls@localunset</code> .....	99, 100

<code>\@glsnextpages</code> .....	120	<code>\@glsxtr@abbreviationsdef</code> .....	18, 26
<code>\@glsnonextpages</code> .....	120	<code>\@glsxtr@accessdisplay</code> .....	179–181
<code>\@glsnumberformat</code> .....		<code>\@glsxtr@activate@initialtagging</code> ..	
..... 9, 11, 66, 68, 84, 148, 179, 182		.....	187, 188
<code>\@glsorder</code> .....	116	<code>\@glsxtr@addunitcounter</code> .....	107
<code>\@glspl@</code> .....	89, 104, 105, 149	<code>\@glsxtr@addunused</code> .....	51
<code>\@glsplural@</code> .....	90	<code>\@glsxtr@addunusedxrefs</code> .....	50, 51
<code>\@glsplunc@token</code> .....	192	<code>\@glsxtr@attrval</code> ....	67, 173–178, 180, 182
<code>\@glsrecordlocref</code> .....	10, 11	<code>\@glsxtr@autoindex@at</code> .....	182–184
<code>\@glsshowtarget</code> .....	88	<code>\@glsxtr@autoindex@doextra@esc</code> ....	182
<code>\@glsstyle@altlist</code> .....	368	<code>\@glsxtr@autoindex@encap</code> .....	182–184
<code>\@glsstyle@altlistgroup</code> .....	369	<code>\@glsxtr@autoindex@esc</code> .....	183, 185, 186
<code>\@glsstyle@altlisthypergroup</code> .....	370	<code>\@glsxtr@autoindex@escat</code> .....	183, 184
<code>\@glsstyle@alttree</code> .....	384	<code>\@glsxtr@autoindex@escencap</code> ...	183, 184
<code>\@glsstyle@alttreegroup</code> .....	395	<code>\@glsxtr@autoindex@esclevel</code> ...	183, 184
<code>\@glsstyle@alttreehypergroup</code> .....	396	<code>\@glsxtr@autoindex@escquote</code> ...	183, 185
<code>\@glsstyle@index</code> .....	379	<code>\@glsxtr@autoindex@level</code> .....	183, 184
<code>\@glsstyle@indexgroup</code> .....	380	<code>\@glsxtr@autoindex@setname</code> .....	182
<code>\@glsstyle@indexhypergroup</code> .....	380	<code>\@glsxtr@autoindex@crossrefs</code> 14, 15, 45, 48	
<code>\@glsstyle@inline</code> .....	378	<code>\@glsxtr@autoseeindexfalse</code> .....	14
<code>\@glsstyle@list</code> .....	368	<code>\@glsxtr@autoseeindextrue</code> .....	16
<code>\@glsstyle@listdotted</code> .....	367	<code>\@glsxtr@bookindex@atendgroup</code> .	404–406
<code>\@glsstyle@listgroup</code> .....	369	<code>\@glsxtr@bookindex@atsubendgroup</code> ..	405
<code>\@glsstyle@listhypergroup</code> .....	369	<code>\@glsxtr@bookindex@atsubsubendgroup</code>	405
<code>\@glsstyle@mcolalttree</code> .....	399	<code>\@glsxtr@bookindex@between</code> ....	404, 406
<code>\@glsstyle@mcolalttreegroup</code> .....	400	<code>\@glsxtr@bookindex@sep</code> .....	404, 405
<code>\@glsstyle@mcolalttreehypergroup</code> ..	400	<code>\@glsxtr@bookindex@subatendgroup</code> ..	
<code>\@glsstyle@mcolalttreesspannav</code> ....	401	.....	404–406
<code>\@glsstyle@mcolindexgroup</code> .....	396	<code>\@glsxtr@bookindex@subbetween</code> .	404, 405
<code>\@glsstyle@mcolindexhypergroup</code> ....	396	<code>\@glsxtr@bookindex@subsep</code> .....	404, 405
<code>\@glsstyle@mcolindexspannav</code> .....	397	<code>\@glsxtr@bookindex@subsubatendgroup</code>	
<code>\@glsstyle@mcoltreegroup</code> .....	397	.....	404–406
<code>\@glsstyle@mcoltreehypergroup</code> .....	398	<code>\@glsxtr@bookindex@subsubbetween</code> ..	
<code>\@glsstyle@mcoltreenamegroup</code> ....	398	.....	404, 405
<code>\@glsstyle@mcoltreenamehypergroup</code>	399	<code>\@glsxtr@bookindex@groupskip</code> ...	404, 406
<code>\@glsstyle@mcoltreenamepannav</code> ..	399	<code>\@glsxtr@cat</code> .....	101, 113, 147, 187
<code>\@glsstyle@mcoltreesspannav</code> .....	398	<code>\@glsxtr@checkgroup</code> .....	140
<code>\@glsstyle@tree</code> .....	380	<code>\@glsxtr@counterrecordhook</code> .....	11
<code>\@glsstyle@treegroup</code> .....	381	<code>\@glsxtr@csname</code> .....	108, 110
<code>\@glsstyle@treehypergroup</code> .....	382	<code>\@glsxtr@current@style</code> .....	57, 401
<code>\@glsstyle@treename</code> .....	382	<code>\@glsxtr@currentunitcount</code> .....	108, 110
<code>\@glsstyle@treenamegroup</code> .....	383	<code>\@glsxtr@currunitcount</code> .....	109, 111
<code>\@glsstyle@treenamehypergroup</code> ...	383	<code>\@glsxtr@debugnr</code> .....	24
<code>\@glstarget</code> .....	89, 122	<code>\@glsxtr@debugval</code> .....	24
<code>\@glstext@</code> .....	90	<code>\@glsxtr@declareoption</code> ...	5, 16, 18, 21, 23
<code>\@glunset</code> .....	98, 99	<code>\@glsxtr@defaultnoglossarywarning</code> ..	21
<code>\@glswidestname</code> .....	386, 394	<code>\@glsxtr@defaultnumberformat</code> .....	
<code>\@glsxtr</code> .....	55, 56	.....	7, 9, 66, 68, 84, 179, 182
<code>\@glsxtr@@do@wrglossary</code> .....	117	<code>\@glsxtr@defpostpunc</code> .....	16, 17, 25

<code>\@glsxtr@deprecated@abbrstyle</code> .....	<code>\@glsxtr@glossnamefont</code> . 175–178, 180, 181
..... 243, 245, 246,	<code>\@glsxtr@gobbleto@endescpch</code> ..... 185
248, 257, 259, 260, 262, 275, 278, 282, 284	<code>\@glsxtr@groupheading</code> ..... 140, 143
<code>\@glsxtr@dialect</code> ..... 36, 37	<code>\@glsxtr@idx@displaynumberlist</code> .... 118
<code>\@glsxtr@disabledflycommand</code> ..... 56	<code>\@glsxtr@idx@entrynumberlist</code> ..... 118
<code>\@glsxtr@display@loc</code> ..... 127	<code>\@glsxtr@ifcsstart</code> ..... 54
<code>\@glsxtr@do@wrindex</code> ..... 85	<code>\@glsxtr@ifpunctoken</code> ..... 193
<code>\@glsxtr@do@glsgldisablehyperinlist</code> .. 83	<code>\@glsxtr@ifunitcounter</code> ..... 107
<code>\@glsxtr@do@inc@linkcount</code> ..... 153	<code>\@glsxtr@insert@dots</code> ..... 194
<code>\@glsxtr@do@record@wrglossary</code> .... 8, 14	<code>\@glsxtr@insert@dots@next</code> ..... 194, 195
<code>\@glsxtr@do@redef@forglsentries</code> ..... 7	<code>\@glsxtr@insertdots</code> ..... 162, 196
<code>\@glsxtr@do@style</code> ..... 22, 330	<code>\@glsxtr@label</code> ..... 32, 51, 153, 172, 173
<code>\@glsxtr@do@titlecaps@warn</code> .....	<code>\@glsxtr@loadstyles</code> ..... 366, 367
..... 174–177, 180, 187	<code>\@glsxtr@local@textformat</code> ..... 66, 67
<code>\@glsxtr@doabbreviationsdef</code> ..... 18	<code>\@glsxtr@locale</code> ..... 36, 37
<code>\@glsxtr@doaccsupp</code> ..... 21, 25	<code>\@glsxtr@longnewglossaryentry</code> ..... 39
<code>\@glsxtr@docdefsetting</code> ..... 15, 52	<code>\@glsxtr@mark@wordseps</code> ..... 195
<code>\@glsxtr@docdefval</code> ..... 15, 51–53	<code>\@glsxtr@mark@wordseps@next</code> ..... 195
<code>\@glsxtr@docounterrecord</code> ..... 11	<code>\@glsxtr@markwordseps</code> ..... 196, 197
<code>\@glsxtr@doglossary</code> ..... 140, 141	<code>\@glsxtr@mixed@assign@sortkey</code> .... 119
<code>\@glsxtr@doiflabelinlist</code> ..... 142	<code>\@glsxtr@noidx@displaynumberlist</code> .. 118
<code>\@glsxtr@dolocktag</code> ..... 59, 60	<code>\@glsxtr@noidx@do</code> ..... 142
<code>\@glsxtr@dorecord</code> ..... 8, 10	<code>\@glsxtr@noidx@entrynumberlist</code> .... 118
<code>\@glsxtr@dorecordnodefer</code> ..... 8, 10	<code>\@glsxtr@noidx@numberlistloop</code> .... 118
<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@dostylewarn</code> ..... 214	<code>\@glsxtr@noop@recordcounter</code> ..... 11, 13
<code>\@glsxtr@enabletagging</code> ..... 186	<code>\@glsxtr@nopostpunc</code> ..... 120
<code>\@glsxtr@end@</code> ..... 54	<code>\@glsxtr@nopostpunc@postdesc</code> ..... 121
<code>\@glsxtr@endescspch</code> ..... 183–186	<code>\@glsxtr@notfoundinlist</code> ..... 193
<code>\@glsxtr@entrycount@org@localreset</code> 102	<code>\@glsxtr@op@recordcounter</code> ..... 14
<code>\@glsxtr@entrycount@org@localunset</code> 102	<code>\@glsxtr@optlist</code> ..... 56
<code>\@glsxtr@entrycount@org@reset</code> ..... 102	<code>\@glsxtr@org@@starttoc</code> ..... 311
<code>\@glsxtr@entrycount@org@unset</code> ..... 102	<code>\@glsxtr@org@GLS@</code> ..... 63
<code>\@glsxtr@entryunitcount@org@localreset</code>	<code>\@glsxtr@org@GLSpl@</code> ..... 63
..... 110	<code>\@glsxtr@org@Gls@</code> ..... 63
<code>\@glsxtr@entryunitcount@org@localunset</code>	<code>\@glsxtr@org@Glspl@</code> ..... 63
..... 110	<code>\@glsxtr@org@Glsxtrtitlefirst</code> . 312, 313
<code>\@glsxtr@entryunitcount@org@reset</code> . 110	<code>\@glsxtr@org@Glsxtrtitlefirstplural</code>
<code>\@glsxtr@entryunitcount@org@unset</code> . 110	..... 312, 314
<code>\@glsxtr@err@undefaction</code> ..... 7, 13	<code>\@glsxtr@org@Glsxtrtitlefull</code> .. 312, 314
<code>\@glsxtr@field@linkdefs</code> ..... 62	<code>\@glsxtr@org@Glsxtrtitlefullpl</code> 312, 314
<code>\@glsxtr@format@overridefalse</code> ..... 181	<code>\@glsxtr@org@Glsxtrtitlelong</code> .. 312, 314
<code>\@glsxtr@format@overridetrue</code> ..... 181	<code>\@glsxtr@org@Glsxtrtitlelongpl</code> 312, 314
<code>\@glsxtr@foundinlist</code> ..... 193	<code>\@glsxtr@org@Glsxtrtitlename</code> .. 312, 313
<code>\@glsxtr@full</code> ..... 200	<code>\@glsxtr@org@Glsxtrtitleplural</code> 312, 313
<code>\@glsxtr@fullpl</code> ..... 202	<code>\@glsxtr@org@Glsxtrtitleshort</code> . 312, 313
<code>\@glsxtr@gettype</code> ..... 119	<code>\@glsxtr@org@Glsxtrtitleshortpl</code> 312, 313
<code>\@glsxtr@glossdescfont</code> ..... 173–175	<code>\@glsxtr@org@Glsxtrtitletext</code> .. 312, 313



<code>\@glxtr@org@MakeUppercase</code> ....	312, 313	<code>\@glxtr@pagetag</code> .....	59, 60
<code>\@glxtr@org@checkfirsthyper</code> ...	83, 115	<code>\@glxtr@prevunitcount</code> .....	109
<code>\@glxtr@org@currentfieldvalue</code> ....	36	<code>\@glxtr@printglossnr</code> .....	121, 122
<code>\@glxtr@org@delimN</code> .....	60	<code>\@glxtr@printglossopts</code> .....	56, 119, 121
<code>\@glxtr@org@delimR</code> .....	60	<code>\@glxtr@printglossval</code> .....	121
<code>\@glxtr@org@doseeglossary</code> ....	25, 117	<code>\@glxtr@printunsrtglossaryskipentry</code> .....	140, 141
<code>\@glxtr@org@glautosee</code> .....	26	<code>\@glxtr@provide@addstoragekey</code> ....	29
<code>\@glxtr@org@glolinkprefix</code> ....	66, 68	<code>\@glxtr@provide@storagekey</code> .....	28
<code>\@glxtr@org@glsl@</code> .....	62	<code>\@glxtr@record</code> .....	13, 14, 62–64, 68, 200, 202–211
<code>\@glxtr@org@glsdohypertarget</code> ....	122	<code>\@glxtr@record@noglossarywarning</code> ..	14
<code>\@glxtr@org@glsignore</code> .....	60	<code>\@glxtr@record@setting</code> .....	8, 10, 13, 47, 52, 53, 116
<code>\@glxtr@org@glspl@</code> .....	62, 63	<code>\@glxtr@record@setting@alsoindex</code> .	8, 10, 47, 116
<code>\@glxtr@org@glxtrtitlefirst</code> .	312, 313	<code>\@glxtr@record@setting@off</code> .....	52
<code>\@glxtr@org@glxtrtitlefirstplural</code> .....	312, 313	<code>\@glxtr@record@setting@only</code> .....	116
<code>\@glxtr@org@glxtrtitlefull</code> ..	312, 314	<code>\@glxtr@recordsee</code> .....	14, 25, 47
<code>\@glxtr@org@glxtrtitlefullpl</code>	312, 314	<code>\@glxtr@redef@forglsentries</code> ..	7, 26, 27
<code>\@glxtr@org@glxtrtitlelong</code> ..	312, 314	<code>\@glxtr@redefstyles</code> .....	22, 330
<code>\@glxtr@org@glxtrtitlelongpl</code>	312, 314	<code>\@glxtr@reg@glosslist</code> ....	116–119, 122
<code>\@glxtr@org@glxtrtitlename</code> ..	312, 313	<code>\@glxtr@restore@postpunc</code> .....	121
<code>\@glxtr@org@glxtrtitleorpdforheading</code> .....	312, 313	<code>\@glxtr@rglstrigger@record</code> ...	149–151
<code>\@glxtr@org@glxtrtitleplural</code>	312, 313	<code>\@glxtr@s@longnewglossaryentry</code> ....	39
<code>\@glxtr@org@glxtrtitleshort</code> .	312, 313	<code>\@glxtr@savepreloctag</code> .....	59–61
<code>\@glxtr@org@glxtrtitleshortpl</code>	312, 313	<code>\@glxtr@setentrycountunsetattr</code> ...	101
<code>\@glxtr@org@glxtrtitletext</code> ..	312, 313	<code>\@glxtr@setentryunitcountunsetattr</code>	112
<code>\@glxtr@org@makeglossaries</code> .....	116	<code>\@glxtr@setupshortcuts</code> .....	20, 21, 26
<code>\@glxtr@org@markboth</code> .....	311	<code>\@glxtr@shortcutsnr</code> .....	20
<code>\@glxtr@org@markright</code> .....	310, 311	<code>\@glxtr@shortcutsval</code> .....	20, 136
<code>\@glxtr@org@newacronymstyle</code> ..	114, 115	<code>\@glxtr@swaptwo</code> .....	193
<code>\@glxtr@org@postdescription</code> ..	121, 188	<code>\@glxtr@tag</code> .....	187
<code>\@glxtr@org@see@noindex</code> .....	134	<code>\@glxtr@taggingcs</code> .....	187
<code>\@glxtr@org@setacronymstyle</code> ..	114, 115	<code>\@glxtr@textformat</code> .....	67
<code>\@glxtr@org@theHvalue</code> .....	8, 9	<code>\@glxtr@theHvalue</code> ...	8–10, 65, 66, 68, 148
<code>\@glxtr@org@unset@buffer</code> .....	98, 99	<code>\@glxtr@thevalue</code> ....	8–10, 64, 66, 68, 148
<code>\@glxtr@org@prefix</code> .....	10, 11	<code>\@glxtr@thisloctag</code> .....	60
<code>\@glxtr@org@printglossary</code> .....	56, 121	<code>\@glxtr@titlelabel</code> .....	124, 125, 142
<code>\@glxtr@org@warndep</code> .....	194	<code>\@glxtr@tmp</code> .....	22, 65, 127, 128
<code>\@glxtr@p@acrlong@</code> .....	90	<code>\@glxtr@type</code> .....	173
<code>\@glxtr@p@acrlongpl@</code> .....	90	<code>\@glxtr@unitcountlist</code> .....	107
<code>\@glxtr@p@acrshort@</code> .....	90	<code>\@glxtr@unset</code> .....	98, 99
<code>\@glxtr@p@acrshortpl@</code> .....	90	<code>\@glxtr@unset@buffer</code> .....	98, 99
<code>\@glxtr@p@long@</code> .....	89	<code>\@glxtr@unsrt@getgroupptitle</code> ....	140
<code>\@glxtr@p@longpl@</code> .....	89	<code>\@glxtr@usesee</code> .....	46
<code>\@glxtr@p@plural@</code> .....	89	<code>\@glxtr@warn@onexistsordo</code> .....	7, 14
<code>\@glxtr@p@short@</code> .....	89	<code>\@glxtr@warn@undefaction</code> .....	7, 14
<code>\@glxtr@p@shortpl@</code> .....	89		
<code>\@glxtr@p@text@</code> .....	89		
<code>\@glxtr@pagetag</code> .....	59, 60		

\@glxstr@wrglossary@locationhyperlink	\@mfu@nocaplist	188
..... 24	\@one	103, 112, 145
\@glxstr@wrglossnr	\@newglossaryentry@defcounters	101, 109
\@glxstr@wrglossval	\@newglossaryentryposthook	.....
\@glxstrbuffer@nodup@unset	..... 12, 13, 28, 48, 81	
\@glxstrbuffer@unset	\@newglossaryentryprehook	.....
\@glxstrdialecthook	..... 12, 13, 28, 39, 40, 48, 81	
\@glxstrdocdeffalse	\@nihil	128, 143
\@glxstrentryfmt	\@nnihil	183, 185, 186, 193–195
\@glxstrfmt	\@no@glxstrindexaliased	84, 85
\@glxstrglossentry	\@no@makeglossaries	133
\@glxstrglossentryother	\@nocounterr	153
\@glxstrhypernameprefix	\@nopostdesc	120
\@glxstrifhasfield	\@onelevel@sanitize	12, 47, 56, 125, 142
\@glxstrifhyphenstart	\@onlypreamble	.....
\@glxstrindexaliased	.. 57, 60, 111, 134, 137, 153, 182, 184–186	
\@glxstrindexcrossrefsfalse	\@org@glossaryentrynumbers	120
\@glxstrindexcrossrefstrue	\@org@newglossaryentryprehook	39, 40
\@glxstrinmark	\@print@unsrt@glossary	139
\@glxstrlong	\@printgloss@setsort	119, 120
\@glxstrlongpl	\@printglossary	56, 139
\@glxstrnewgls	\@printunsrt@glossary@handler	141
\@glxstrnewgls@inner	\@printunsrtglossary	139
\@glxstrnewgls@innercsname	\@rGLS	150
\@glxstrnotinmark	\@rGLS@	150
\@glxstrp	\@rGLSpl	151
\@glxstrp@opt	\@rGLSpl@	151
\@glxstrpl	\@rGls	149
\@glxstrpostloctag	\@rGls@	150
\@glxstrpreloctag	\@rGlspl	150
\@glxstrsetaliasnoindex	\@rGlspl@	150
\@glxstrshort	\@rgls	149
\@glxstrshortpl	\@rgls@	149
\@glxstrundeftag	\@rglspl	149
\@glxstrwrglossmark	\@rglspl@	149
\@gobble .. 7, 13, 16, 69, 141, 142, 194, 404–406	\@sGlsXtrEnableOnTheFly	54
\@gobbletwo	\@secondofthree	70–
\@ifnextchar	72, 77–80, 82, 201, 203, 205, 206, 208, 210	
\@ifpackageloaded	\@secondoftwo	63, 69,
154, 173, 175, 177, 179, 181, 330, 334, 365	72–80, 83, 89, 115, 122, 179, 193, 200–	
\@ifstar	202, 204–211, 225, 247, 261, 283, 312, 313	
29, 32, 39, 40, 42, 54, 86, 98, 99, 139, 186	\@sglsxtr@provide@storagekey	28
\@ifundefined	\@starttoc	311
330	\@thirdofthree	70–72,
\@ignored@glossaries	77–80, 82, 202, 203, 205, 207, 209, 211, 313	
41–43	\@thirdoftwo	72–76
\@input	\@this@key	179
134	\@tracklang@lang	37
\@input@	\@warn@nomakeglossaries	130
129		
\@istfilename		
116		
\@makeglossary		
116		
\@mfu@domakefirstuc		
187, 188		

\@xdy@main@language	129	\acl	19
\@xdycrossrefhook	47	\ACLP	19
\@xdy@language	129	\Aclp	19
\@xdy@locationclassorder	47	\aclp	19
\\	128	\ACP	19
		\Acp	19
		\acp	19
\_	53, 131, 132	\ACRfullfmt	114
		\Acrfullfmt	114
		\acrfullfmt	114
		\ACRfullplfmt	114
		\Acrfullplfmt	114
		\acrfullplfmt	114
		\acronymentry	113
		\acronymfont	77–80, 92, 114, 115
		\acronymname	17
		\acronymsort	113
		\acronymtype	18, 113, 114
		\acrpluralsuffix	113, 136
		\ACS	19
		\Acs	19
		\acs	19
		\ACSP	19
		\Acsp	19
		\acsp	19
		\actualchar	185
		\addtolength	395
		\advance	103, 112, 135, 145
		\AF	18
		\Af	18
		\af	18
		\AFP	18
		\Afp	18
		\afp	18
		\AL	18
		\Al	18
		\al	18
		\ALP	18
		\Alp	18
		\alp	18
		amsmath package	23
		\AnyTrackedLanguages	330, 365
		\appto	12, 13, 22, 28, 45, 47–49, 81, 85, 101, 109, 140–142, 181, 192, 195, 366
		\arabic	23, 406
		\AS	18
		\As	18
		\as	18
		\ASP	18
<b>A</b>			
\AA	348		
\aa	348		
\AB	18		
\Ab	18		
\ab	18		
abbreviation styles:			
long-hyphen-postshort-hyphen	299, 301		
long-hyphen-short-hyphen	295, 300		
long-postshort-user	288		
long-short-user	286		
nolong-short	230		
short	228		
short-hyphen-long-hyphen	303, 305		
short-hyphen-postlong-hyphen	304, 307		
short-long-user	288		
short-nolong	227, 230		
short-nolong-desc	229		
short-postlong-user	290		
\abbreviationsname	17		
\abbrvpluralsuffix			
.	136, 162, 163, 197, 218, 220, 223, 225, 226, 228, 231, 235, 237, 238, 240, 242, 243, 245, 247, 249, 251, 252, 254, 256, 257, 259, 261, 263, 265, 267, 268, 270, 271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 292, 295, 297, 300, 303, 305, 308		
\ABP	18		
\Abp	18		
\abp	18		
\AC	19		
\Ac	19		
\ac	19		
\ACF	19		
\Acf	19		
\acf	19		
\ACFP	19		
\Acfp	19		
\acfp	19		
\ACL	19		
\Acl	19		

<code>\Asp</code> .....	18	<code>\cGLs</code> .....	18, 19, 101, 112
<code>\asp</code> .....	18	<code>\cglS</code> .....	18, 19, 101, 112
<code>\AtBeginDocument</code> .....	24, 27, 58, 136	<code>\cGLSformat</code> .....	105
<code>\AtEndDocument</code> .....	50, 102, 111, 129, 130	<code>\cGLSformat</code> .....	104
<b>B</b>			
<code>babel</code> package .....	182, 184, 192	<code>\cGLSpl</code> .....	18, 19, 101, 112
<code>\begin</code> .....	126, 131, 132, 140, 368, 370–377, 380, 382, 397–401, 404	<code>\cGlSpl</code> .....	18, 19, 101, 112
<code>\begingroup</code> .....	8, 9, 30, 84, 138, 139, 152	<code>\cglSpl</code> .....	18, 19, 101, 112
<code>\bgroup</code> .....	39, 40, 120	<code>\cGLSplformat</code> .....	105
<code>bib2gls</code> ...	23, 30, 143, 148, 149, 330, 331, 333	<code>\cGlSplformat</code> .....	104
<b>C</b>			
<code>\c@wrglossary</code> .....	23	<code>\cglSplformat</code> .....	104, 106
<code>\catcode</code> .....	51, 134	<code>\changes</code> .....	316, 380, 382
category attributes:		<code>\char</code> .....	125
<code>accessinsertdots</code> .....	162	<code>\columnwidth</code> .....	58
<code>apospplural</code> .....	197	<code>\count@</code> .....	103, 111, 112
<code>discardperiod</code> .....	191	<code>\csappto</code> .....	38
<code>entrycount</code> .....	98, 100, 101, 103, 112	<code>\csdef</code> .....	28, 34, 38, 39, 81, 82, 102, 107, 108, 110, 168, 179, 189, 190, 215, 216, 224, 247, 261, 282, 286, 288–290, 300, 302, 305, 307, 367, 385
<code>firstshortaccess</code> .....	164	<code>\cseappto</code> .....	43
<code>firstuc</code> .....	177	<code>\csedef</code> .....	108
<code>glossdesc</code> .....	173	<code>\csgdef</code> .....	35, 41–43, 53, 59, 102, 108, 110, 111, 385, 407
<code>glossdescfont</code> .....	173	<code>\cslet</code> .....	34, 40, 120
<code>glossname</code> .....	175	<code>\csletcs</code> .....	34, 216
<code>glossnamefont</code> .....	175, 177	<code>\csname</code> .....	6, 29, 42, 47, 52, 57, 61, 63, 66, 68, 77–82, 84, 93, 108, 117, 126, 129, 132, 133, 140, 145–148, 153, 173, 194, 201–211, 217, 394
<code>headuc</code> .....	314	<code>\cspreto</code> .....	39
<code>indexname</code> .....	182	<code>\csuse</code> .....	9, 30, 31, 42, 50, 60, 81, 82, 94–96, 107–111, 119, 124, 126, 127, 137, 138, 142, 143, 168, 179, 189, 190, 215, 216, 385, 386, 388, 389
<code>indexonlyfirst</code> .....	85	<code>\csxdef</code> .....	45, 48, 108, 111
<code>insertdots</code> .....	196	<code>\currentglossary</code> .....	120, 138, 139, 406
<code>linkcount</code> .....	152	<code>\CurrentOption</code> .....	24, 366, 367
<code>linkcountmaster</code> .....	152	<code>\CurrentTrackedLanguage</code> .....	364
<code>markshortwords</code> .....	196	<code>\CurrentTrackedLanguageTag</code> .....	136
<code>markwords</code> .....	196, 197, 293, 295, 303	<code>\CurrentTrackedScript</code> .....	364, 365
<code>nameshortaccess</code> .....	163	<code>\CurrentTrackedTag</code> .....	330, 365
<code>nohyper</code> .....	83	<code>\CustomAbbreviationFields</code> .....	
<code>nohyperfirst</code> .....	70–72	....	198, 218–222, 224, 226, 228, 231, 233–239, 241, 245, 246, 249–252, 254, 256, 259, 261, 263, 264, 266–271, 273, 275, 278, 280, 282, 285, 286, 288, 290– 293, 295–298, 300, 301, 303–305, 307–309
<code>noshortplural</code> .....	197		
<code>regular</code> .....	61, 106, 217–222, 224, 227, 229, 230, 232– 234, 236, 237, 239, 240, 243–245, 247, 250–254, 257–259, 261, 264–266, 268, 269, 271, 272, 274, 276, 278, 280–282, 285, 291–293, 295–298, 303, 304, 308, 310		
<code>textformat</code> .....	67		
<code>textshortaccess</code> .....	163		
<code>\cdot</code> .....	24		
<code>\centering</code> .....	403		
<code>\cGLS</code> .....	18, 19, 101, 112		

## D

`\DeclareAcronymList` ..... 113  
`\DeclareOption` ..... 5, 366  
`\DeclareOptionX` ..... 5, 24  
`\def` ..... 9, 10, 12–15, 25, 27,  
     30, 33, 40, 42, 46–48, 51, 54–56, 59, 61–  
     66, 68–80, 86, 88–92, 98, 104–106, 113,  
     117, 119–123, 125–129, 140, 142, 143,  
     145, 148–151, 161–163, 183–188, 192–  
     197, 200–211, 214, 294, 296, 299, 302,  
     304, 305, 365, 378, 379, 394–396, 400, 401  
`\defglentryfmt` ..... 41–43  
`\define@boolkey` ..... 15, 16, 65, 83  
`\define@choicekey` .....  
     ..... 7, 13, 15, 16, 20, 21, 24, 64, 121  
`\define@key` ..... 12, 13,  
     16, 22, 28, 48, 64, 65, 68, 81, 122, 161, 194  
`\DefineAcronymSynonyms` ..... 20  
`\delimN` ..... 60  
`\delimR` ..... 60  
`\detokenize` ..... 54  
`\dimen@` ..... 115, 385–393  
`\dimen@i` ..... 388, 389  
`\dimen@ii` ..... 385, 386, 388, 389  
`\dimexpr` ..... 58, 384  
`\disable@keys` ..... 17, 27, 53, 134  
`\do` ..... 6, 22, 32, 51, 69, 100, 101, 113,  
     116, 119, 126, 140, 147, 153, 172, 179, 187  
`\do@gl@link@checkfirsthyper` .....  
     ..... 30, 62–64, 66, 77–80, 200–211  
`\do@gl@disablehyperinlist` ..... 66, 84  
`doc package` ..... 185  
`\dolistcsloop` ..... 31  
`\DTLifinlist` ..... 117, 118, 122  
`\DTLifint` ..... 125

## E

`\eappto` .....  
     11, 22, 41–43, 141, 143, 162–164, 182, 366  
`\edef` ..... 6, 8–11, 36, 41–  
     44, 47, 49–52, 66–68, 83, 84, 87, 88, 107,  
     108, 110, 116, 117, 122, 125, 127–130,  
     134, 138, 139, 148, 152, 173–176, 178–  
     180, 183, 185, 194, 364, 388, 389, 404, 405  
`\eglssetwidest` ..... 387–393  
`\egroup` ..... 40, 120  
`\else` 8–12, 15, 16, 18, 20, 21, 25, 26, 30, 33,  
     38, 51, 53, 54, 59, 61, 63, 67, 84, 85, 103,  
     115, 116, 118, 120–122, 125, 127, 128,

    131, 133, 135, 148, 181–183, 185, 193–  
     195, 197, 204–211, 213, 214, 218, 220,  
     221, 223–232, 235, 237–251, 253–265,  
     267, 269–284, 286, 287, 289, 290, 292–  
     294, 296, 299–302, 305, 306, 308, 309,  
     368, 370–381, 383, 394, 395, 401, 403, 405  
`\emph` ..... 262, 263  
`\empty` ..... 127, 128  
`\encapchar` ..... 185  
`\end` ..... 122, 126,  
     131, 132, 141, 368, 370–377, 397–401, 404  
`\end@gl@xtr@display@loc` ..... 127  
`\endcsname` ..... 6, 29, 42,  
     47, 52, 57, 61, 63, 66, 68, 77–82, 84, 93,  
     108, 117, 126, 129, 130, 132, 133, 140,  
     145–148, 153, 173, 194, 201–211, 217, 394  
`\endgroup` ..... 8, 10, 30, 84, 138, 139, 152  
`\ensuremath` ..... 24  
entry categories:  
    abbreviation ..... 211  
    general ..... 167, 169  
    index ..... 171  
`\epreto` ..... 182  
`\equal` ..... 133, 190  
`etoolbox package` ..... 5  
`\expandafter` ..... 24, 29, 30, 32, 36, 45,  
     46, 50, 51, 54–56, 65, 81, 82, 86, 93, 99,  
     106, 107, 117–119, 122, 126–128, 140,  
     143, 145, 152, 162, 163, 173, 176, 177,  
     179, 181, 182, 185, 193, 196, 197, 293, 404  
`\expandonce` . 113, 143, 162–164, 183, 219, 297  
`\ExtraCustomAbbreviationFields` ....  
     ..... 162–164, 196, 198

## F

`\fi` ..... 7–12, 14–16, 18, 20, 21,  
     24–26, 30, 33, 38, 45, 47, 48, 50, 52–54,  
     57–59, 61, 64, 67, 84, 85, 103, 111, 112,  
     115, 118–122, 125, 127, 128, 130, 131,  
     133, 135, 136, 148, 181–183, 186, 193,  
     195, 197, 204–211, 213, 214, 218, 220,  
     221, 223–232, 235, 237–251, 253–265,  
     267, 269–284, 286, 287, 289, 290, 292,  
     294, 296, 299–302, 305, 306, 308, 309,  
     368, 371–381, 383, 385–395, 401, 403, 406  
first use ..... 408  
    flag ..... 408  
    text ..... 408  
`\firstacronymfont` ..... 114, 115  
`fontspec package` ..... 136

`\footnote` ..... 222  
`\forall glossaries` 50, 140, 171, 173, 387–393  
`\forall glossentries` ..... 52, 103, 112  
`\forall TrackedDialect` ..... 330, 365  
`\foreignlanguage` ..... 36, 37  
`\forall glossentries` ..... 6, 50, 171, 173, 387–393  
`\forall listcsloop` ..... 32, 112, 126  
`\forall listloop` ..... 99, 123, 124, 188  
`\futurelet` ..... 192

## G

`\gdef` ..... 60, 184, 185  
`\Genacrfullformat` ..... 114  
`\genacrfullformat` ..... 114  
`\GenericAcronymFields` ..... 114  
`\Genplacrfullformat` ..... 114  
`\genplacrfullformat` ..... 114  
`\GetTrackedDialectFromLanguageTag` .. 36  
`\GetTrackedDialectToMapping` ..... 36  
`\glo@grabfirst` ..... 143  
`\glo@name` ..... 176, 177, 181  
`\gloaliaslabel` ..... 88  
`\global` ..... 10, 40, 52, 120, 143  
`\glo linkprefix` ..... 65–68, 88, 122, 136  
glossaries package .....  
..... 14, 25, 26, 38, 45, 47–49, 119, 367  
glossaries-accsupp package .... 21, 25, 154, 197  
glossaries-extra package ..... 2, 365  
glossaries-extra-bib2gls package 14, 27, 330, 365  
glossaries-extra-stylemods package . 21, 189, 330  
glossaries-stylemods package ..... 402  
glossaries.sty package ..... 40  
`\GlossariesExtraWarning` ..... 6,  
16, 21, 36–39, 54, 56, 67, 115, 117, 127,  
131, 134, 140, 173–176, 178, 180, 187, 216  
`\GlossariesExtraWarningNoLine` 16, 103, 112  
`\GlossariesWarning` 59, 118, 120, 123, 124, 214  
`\GlossariesWarningNoLine` ..... 117, 130  
glossary styles:  
altlist ..... 368  
altlistgroup ..... 369  
altlisthypergroup ..... 370  
alttree ..... 384, 385, 394  
alttreegroup ..... 395  
alttreehypergroup ..... 396  
index ..... 379  
indexgroup ..... 380  
indexhypergroup ..... 380  
inline ..... 378

list ..... 368  
listdotted ..... 367  
listdottedstyle ..... 368  
listgroup ..... 369  
listhypergroup ..... 369  
mcolalttree ..... 399  
mcolalttreegroup ..... 400  
mcolalttreehypergroup ..... 400  
mcolalttreespannav ..... 401  
mcolindexgroup ..... 396  
mcolindexhypergroup ..... 396  
mcolindexspannav ..... 397  
mcoltreegroup ..... 397  
mcoltreehypergroup ..... 398  
mcoltreenonamegroup ..... 398  
mcoltreenonamehypergroup ..... 399  
mcoltreenonamespannav ..... 399  
mcoltreespannav ..... 398  
sublistdotted ..... 368  
tree ..... 380  
treegroup ..... 381  
treehypergroup ..... 382  
treenoname ..... 382  
treenonamegroup ..... 383  
treenonamehypergroup ..... 383  
glossary-bookindex package ..... 367  
glossary-hypernav package ..... 87  
glossary-long package ..... 372  
glossary-longbooktabs package ..... 372  
glossary-tree package ..... 378, 379  
`\glossaryentrynumbers` ..... 61, 120, 144  
`\glossaryheader` ..... 126,  
140, 368–377, 379–383, 394, 396–400, 404  
`\glossaryname` ..... 119  
`\glossarypostamble` ..... 126, 141, 142  
`\glossarypreamble` ..... 126, 140  
`\glossarysection` ... 126, 132, 134, 140, 142  
`\glossarytitle` 42, 119, 120, 126, 132, 134, 140  
`\glossarytoctitle` 42, 119, 126, 132, 134, 140  
`\glossentry` .....  
120, 144, 367–377, 379, 381, 383, 394, 404  
`\glossentrydesc` .....  
..... 367, 368, 370–377, 380–382, 384  
`\glossentryname` .....  
138, 367–377, 379, 381, 383, 394, 395, 402  
`\glossentrynameother` ..... 139  
`\glossentrysymbol` 371–375, 377, 381, 382, 384  
`\glossxtrsetpopts` ..... 189  
`\glostyle` ..... 380, 382

<code>\GLS</code> .....	101, 112, 147	<code>\Glsaccessfirst</code> .....	70
<code>\Gls</code> .....	55, 101, 112, 147	<code>\glsaccessfirst</code> .....	70
<code>\gls</code> .....	38, 55, 57, 101, 112, 118, 131, 147	<code>\GLSaccessfirstplural</code> .....	72
<code>\gls@assign@desc</code> .....	40	<code>\Glsaccessfirstplural</code> .....	72
<code>\gls@assign@field</code> .....	12, 13, 28, 81	<code>\glsaccessfirstplural</code> .....	71
<code>\gls@checkseeallowed</code> .....	52–54, 117	<code>\Glsaccesslong</code> .....	79,
<code>\gls@codepage</code> .....	130		198, 206, 218, 227, 231, 232, 235, 241,
<code>\gls@defdocnewglossaryentry</code> .	51, 101, 109		242, 244, 249, 255–258, 264, 265, 273–
<code>\gls@defglossaryentry</code> .....	40, 52, 55, 56		279, 286, 287, 295, 297, 298, 301, 303, 309
<code>\gls@dotocitle</code> .....	120	<code>\glsaccesslong</code> ...	79, 198, 206, 207, 218,
<code>\gls@glossary</code> .....	47		220, 221, 223–226, 228–232, 235, 237–
<code>\gls@grplabel</code> .....	87		246, 248, 249, 251, 253–260, 262, 263,
<code>\gls@ifnotmeasuring</code> .....	100		265, 267, 269, 270, 272–283, 286, 287,
<code>\gls@level</code> .....	143, 144		290, 292, 295, 297, 298, 301, 303, 308, 309
<code>\gls@noidxglossary</code> .....	117	<code>\Glsaccesslongpl</code> .....	
<code>\gls@org@glossaryentryfield</code> .....	120		.. 80, 199, 210, 218, 227, 232, 235, 241,
<code>\gls@org@glossarysubentryfield</code> ...	120		242, 244, 250, 255–258, 264, 265, 273–
<code>\gls@orgTrackLangRequireDialectPrefix</code>			279, 286, 287, 295, 297, 298, 301, 303, 309
	364, 365	<code>\glsaccesslongpl</code> .....	80,
<code>\gls@save@numberlist</code> .....	59, 61		199, 210, 211, 218, 221, 223–232, 235,
<code>\gls@set@xr@key</code> .....	48		237–246, 248, 249, 251, 253–260, 262,
<code>\gls@tmplen</code> .....	387–393, 395		263, 265, 267, 269–284, 286, 287, 290,
<code>\gls@type</code> .....	117		292, 293, 295, 297, 298, 301, 303, 308, 309
<code>\glsabbrvdefaultfont</code> ...	200, 218, 220,	<code>\GLSaccessname</code> .....	72
	223, 225, 226, 228, 231, 284, 294, 297, 307	<code>\Glsaccessname</code> .....	72
<code>\glsabbrvfont</code> .....		<code>\glsaccessname</code> .....	46, 72
	262–271, 273, 275, 277, 279, 281–283	<code>\GLSaccessplural</code> .....	71
<code>\glsabbrvfont</code> .....		<code>\Glsaccessplural</code> .....	71
	90, 91, 114, 200, 204, 205, 208, 209,	<code>\glsaccessplural</code> .....	71
	211, 213, 214, 217–226, 228, 231, 233,	<code>\Glsaccessshort</code> .....	77, 205, 214, 221,
	235, 237, 238, 240, 242, 243, 245, 247,		223–225, 229, 230, 237, 239, 240, 246–
	249, 251, 252, 254, 256, 257, 259, 261,		248, 251, 253, 254, 260, 262, 267, 269,
	263, 265, 267, 268, 270, 271, 273, 275,		270, 272, 281–283, 289, 290, 292, 301, 306
	277, 279, 281, 283, 285, 287, 289, 291–	<code>\glsaccessshort</code> ..	77, 198, 199, 204, 205,
	293, 295, 297, 299–301, 303, 305, 306, 308		213, 218, 220, 223, 225–230, 232, 235,
<code>\glsabbrvhyphenfont</code> .....			237–242, 244–249, 251–265, 267, 269–
	294–296, 300, 302–305, 307		274, 276–279, 281, 283, 286, 287, 289,
<code>\glsabbrvonlyfont</code> .....	307, 308, 310		290, 292, 295, 297, 298, 300, 303, 306, 309
<code>\glsabbrvscfont</code> .	234–240, 242, 243, 245–247	<code>\Glsaccessshortpl</code>	78, 208, 214, 221, 223–
<code>\glsabbrvsmfont</code> .....			226, 229, 230, 237, 239, 240, 246, 248,
	248–252, 254, 256, 257, 259, 261		251, 253, 254, 260, 262, 267, 269, 270,
<code>\glsabbrvuserfont</code> .....	284–290, 292		272, 281–284, 289, 290, 292, 301, 306, 309
<code>\GLSaccessdesc</code> .....	73	<code>\glsaccessshortpl</code> .....	
<code>\Glsaccessdesc</code> .....	73, 174, 186		78, 199, 208, 209, 214, 218–
<code>\glsaccessdesc</code> .....	73, 174, 190, 191		220, 223, 225–230, 232, 235, 237–251,
<code>\GLSaccessdescplural</code> .....	73		253–255, 257, 258, 260–265, 267, 269–
<code>\Glsaccessdescplural</code> .....	73		274, 276, 278, 279, 281–283, 286, 287,
<code>\glsaccessdescplural</code> .....	73		289, 290, 292, 295, 298, 300, 303, 306, 309
<code>\GLSaccessfirst</code> .....	71	<code>\GLSaccesssymbol</code> .....	74



<code>\Glsaccesssymbol</code> .....	74, 186	<code>\glsdonohyperlink</code> .....	67, 88, 89
<code>\glsaccesssymbol</code> .....	74, 186, 191	<code>\glsdosanitizesort</code> .....	118
<code>\GLSaccesssymbolplural</code> .....	74	<code>\glsenableentrycount</code> .....	101, 103, 111
<code>\Glsaccesssymbolplural</code> .....	74	<code>\glsenableentryunitcount</code> .....	102, 112
<code>\glsaccesssymbolplural</code> .....	74	<code>\glsentrycounter</code> .....	24, 128
<code>\GLSaccesstext</code> .....	70	<code>\GlsEntryCounterLabelPrefix</code> .....	38
<code>\Glsaccesstext</code> .....	70	<code>\glsentrycurrcount</code> .....	102, 103, 109
<code>\glsaccesstext</code> .....	46, 69	<code>\Glsentrydesc</code> .....	158, 166, 174
<code>\glsacrshortcutstrue</code> .....	20, 21	<code>\glsentrydesc</code> .....	158, 166, 175
<code>\glsacspacemax</code> .....	115	<code>\Glsentrydescplural</code> .....	159, 166
<code>\glsadd</code> .....	30, 51, 69, 131	<code>\glsentrydescplural</code> .....	158, 159, 166
<code>\glsadd options</code>		<code>\Glsentryfirst</code> .....	106, 151, 156, 165
<code>theHvalue</code> .....	9	<code>\glsentryfirst</code> .....	106, 151, 156, 165, 326
<code>thevalue</code> .....	9, 10	<code>\Glsentryfirstplural</code> ...	107, 152, 156, 165
<code>\glsaddpostsetkeys</code> .....	10, 68	<code>\glsentryfirstplural</code> .....	106, 151, 156, 157, 165, 327
<code>\glsaddpresetkeys</code> .....	10, 68	<code>\glsentryfmt</code> .....	41–43
<code>\glsaddstoragekey</code> .....	49, 167, 332, 333	<code>\Glsentryfull</code> .....	114
<code>\glsbackslash</code> .....	54	<code>\glsentryfull</code> .....	114
<code>\glscapscase</code> .....	63, 69–80, 82, 201–213	<code>\Glsentryfullpl</code> .....	114
<code>\glscategory</code> ..	61, 69, 83, 90, 91, 168–170, 173–180, 186, 189, 190, 200–205, 208, 209	<code>\glsentryfullpl</code> .....	114
<code>\glscategorylabel</code> .....	83, 162–164, 194, 196, 197, 224, 247, 261, 282, 286, 288–290, 300, 302, 305, 307	<code>\glsentryitem</code> .....	138, 139, 367–377, 379, 381, 383, 394, 405
<code>\glsclsebrace</code> .....	47, 132, 133	<code>\Glsentrylong</code> .....	91, 92, 106, 151, 160, 167
<code>\glscounter</code> .....	9	<code>\glsentrylong</code> .....	91, 92, 106, 151, 160, 167, 224, 247, 261, 282, 289, 290, 305, 327
<code>\glscurrententrylabel</code> .....	59, 60, 120, 129, 138–141, 188, 189	<code>\Glsentrylongpl</code> .....	91, 92, 107, 161, 167
<code>\glscurrentfieldvalue</code> .....	30–33, 35, 36, 284, 331, 332	<code>\glsentrylongpl</code> ...	91, 92, 106, 161, 167, 328
<code>\glscustomtext</code> ..	62, 63, 77–80, 201–211, 213	<code>\Glsentrylongplural</code> .....	152
<code>\glsdefaulttype</code>	6, 17, 38, 39, 119, 131, 139, 142	<code>\glsentrylongplural</code> .....	151
<code>\glsdescriptionaccessdisplay</code> ..	158, 174	<code>\Glsentryname</code> .....	154, 164, 175, 177, 178
<code>\glsdescriptionpluralaccessdisplay</code>	158, 159	<code>\glsentryname</code> .....	137, 138, 154, 164, 182, 324, 325, 387–393, 407
<code>\glsdescwidth</code> .....	370–377	<code>\glsentrynumberlist</code> ....	118, 124, 391–393
<code>\glsdetoklabel</code> .....	8, 9, 28, 31–35, 38– 40, 44–46, 50, 52–54, 66, 68, 84, 88, 102, 107–112, 117, 120, 123, 124, 138, 139, 143, 146–148, 173, 176, 177, 181, 388, 389	<code>\Glsentryplural</code> .....	155, 165
<code>\glsdisplaynumberlist</code> .....	118, 123	<code>\glsentryplural</code> ....	155, 164, 165, 325, 326
<code>\glsdohyperlink</code> .....	86, 87, 89	<code>\glsentryprevcount</code> .....	102, 103, 109
<code>\glsdohypertarget</code> .....	89, 122	<code>\glsentryprevmaxcount</code> .....	110
<code>\glsdoifexists</code> .....	15, 25, 34, 39, 43, 45–47, 62, 63, 68, 77–80, 88, 100, 117, 123, 124, 138, 139, 200–211	<code>\glsentryprevtotalcount</code> .....	109
<code>\glsdoifexistsordo</code> .....	30, 31, 64	<code>\Glsentryshort</code> .....	90, 92, 159, 166
<code>\glsdoifexistsorwarn</code> ....	15, 173, 174, 186	<code>\glsentryshort</code> .....	90– 92, 115, 159, 166, 286, 288, 299, 323, 324
<code>\glsdoifnoexists</code> .....	39, 40	<code>\Glsentryshortpl</code> .....	91, 92, 160, 166
		<code>\glsentryshortpl</code> .....	91, 92, 160, 166, 167, 323, 324
		<code>\Glsentrysymbol</code> .....	157, 165
		<code>\glsentrysymbol</code> .....	157, 165, 390–392
		<code>\Glsentrysymbolplural</code> .....	158, 165
		<code>\glsentrysymbolplural</code> ..	157, 158, 165, 166



<code>\Glsentrytext</code> .....	155, 164	<code>\glsfirstlongfootnotefont</code> .....	
<code>\glsentrytext</code> .....	88, 154, 155, 164, 325	.....	222–226, 245–248, 259–262, 280–284
<code>\glsentrytype</code> .....	138	<code>\glsfirstlonghyphenfont</code> .....	294–305
<code>\Glsentryuseri</code> .....	75	<code>\glsfirstlongonlyfont</code> .....	308–310
<code>\glsentryuseri</code> .....	75	<code>\glsfirstlonguserfont</code> .....	285–293
<code>\Glsentryuserii</code> .....	75	<code>\GLSfirstplural</code> .....	319
<code>\glsentryuserii</code> .....	75	<code>\Glsfirstplural</code> .....	319
<code>\Glsentryuseriii</code> .....	75	<code>\glsfirstplural</code> .....	319
<code>\glsentryuseriii</code> .....	75	<code>\glsfirstpluralaccessdisplay</code> ..	156, 157
<code>\Glsentryuseriv</code> .....	76	<code>\glsforeachincategory</code> .....	214
<code>\glsentryuseriv</code> .....	76	<code>\glsgenentryfmt</code> .....	61
<code>\Glsentryuserv</code> .....	76	<code>\glsgetattribute</code> .....	67, 87, 103, 107–
<code>\glsentryuserv</code> .....	76	.....	109, 129, 148, 152, 173–176, 178, 180, 182
<code>\Glsentryuservi</code> .....	76	<code>\glsgetcategoryattribute</code> .....	168
<code>\glsentryuservi</code> .....	76	<code>\glsgetgrouptitle</code> .....	
<code>\glsextrapostnamehook</code> .....	179	.....	369, 370, 380, 382–384, 395–401
<code>\glsfieldfetch</code> .....	88	<code>\glsgetwidestname</code> .....	385
<code>\glsfieldxdef</code> .....	172, 173	<code>\glsgroupheading</code> .....	
<code>\glsfindwidesttoplevelname</code> .....	386	.....	143, 368–377, 379–384, 394–401, 406
<code>\GLSfirst</code> .....	318	<code>\glsgroupskip</code> .....	
<code>\Glsfirst</code> .....	319	.....	143, 368, 370–378, 380, 381, 383, 395, 406
<code>\glsfirst</code> .....	318	<code>\glshasattribute</code> .....	
<code>\glsfirstabbrvdefaultfont</code> .....		.....	67, 87, 103, 108, 110, 112, 129,
.....	200, 218, 220, 223, 225, 226, 228, 231, 297	.....	148, 152, 173–178, 180, 182, 218–222,
<code>\glsfirstabbrvmfont</code> .....	263–284	.....	224, 228–230, 233–236, 238, 245, 247,
<code>\glsfirstabbrvfont</code> .....	114, 198, 199,	.....	249–252, 259, 261, 263–266, 268, 269,
.....	218–232, 235, 237, 238, 240, 242, 243,	.....	277, 280–282, 285, 286, 288, 289, 291–
.....	245, 247, 249, 251, 252, 254, 256, 257,	.....	293, 295–298, 300, 302–305, 307, 308, 310
.....	259, 261, 263, 265, 267, 268, 270, 271,	<code>\glshascategoryattribute</code> .....	169
.....	273, 275, 277, 279, 281, 283, 285, 287,	<code>\glshex</code> ..	335–341, 343–349, 351–354, 356–364
.....	289, 292, 295, 297, 298, 300, 303, 305, 308	<code>\glshyperlink</code> .....	88, 332
<code>\glsfirstabbrvhyphenfont</code> .....		<code>\glshypernavsep</code> .....	126
.....	294–296, 299, 300, 302–307	<code>\glshypernumber</code> .....	129, 181
<code>\glsfirstabbrvonlyfont</code> .....	308, 309	<code>\glsifattribute</code> .....	
<code>\glsfirstabbrvscfont</code> .....	234–248	.....	64, 65, 70, 83, 85, 94, 152, 171,
<code>\glsfirstabbrvsmfont</code> .....	249–262	.....	174–177, 180, 181, 188, 191, 192, 314–323
<code>\glsfirstabbrvuserfont</code> .....	285–293	<code>\glsifcategory</code> .....	171
<code>\glsfirstaccessdisplay</code> .....	156	<code>\glsifcategoryattribute</code> .....	
<code>\glsfirstlongdefaultfont</code> .....		.....	83, 162–164, 169, 170, 196, 197
.....	218, 220, 226, 228, 231, 234–244, 249–	<code>\glsifnotregular</code> .....	69
.....	259, 263, 264, 266, 267, 270–275, 277, 278	<code>\glsifnotregularcategory</code> .....	170
<code>\glsfirstlongemfont</code> .....		<code>\glsifplural</code> .....	
.....	265, 266, 268, 269, 275, 276, 278–280	.....	63, 69, 71–74, 77–80, 191, 192, 200–212
<code>\glsfirstlongfont</code> ...	198, 199, 218–221,	<code>\glsifregular</code> .....	61, 69, 106, 107, 151, 152
.....	223, 225–233, 235, 237, 238, 240, 242,	<code>\glsifregularcategory</code> .....	170
.....	243, 245, 247, 249, 251, 252, 254, 256,	<code>\glsifusetranslator</code> .....	42
.....	257, 259, 261, 263, 265, 267, 268, 270,	<code>\glsignore</code> .....	60
.....	271, 273, 275, 277, 279, 281, 283, 285,	<code>\glsinlinedescformat</code> .....	378
.....	287, 289, 292, 295, 297, 300, 303, 305, 308	<code>\glsinlinesubdescformat</code> .....	378

<code>\glsinsert</code> .....	63,	<code>\glslongfootnotefont</code> .....	222, 223, 225, 245, 247, 259, 261, 281, 283
69, 77–80, 201–213, 293, 300, 302, 305, 307		<code>\glslonghyphenfont</code> .....	294, 295, 297, 298, 300, 303, 305
<code>\glskeylisttok</code> .....	113, 114, 195, 198	<code>\glslongonlyfont</code> .....	308
<code>\glslabel</code> .....	8, 9, 30, 44,	<code>\glslongpltok</code> .....	197, 198, 218–222, 231, 233, 234,
46, 61, 64–67, 83, 84, 87, 88, 115, 148,		236, 237, 241, 245, 249–252, 256, 259,	
152, 153, 189–191, 211–213, 224, 247,		263–269, 273, 275, 278, 281, 285, 286,	
261, 282, 286, 288–290, 300, 302, 305, 307		288–293, 295–298, 300, 302–304, 308, 310	
<code>\glslabeltok</code> ....	113, 195, 198, 218–224,	<code>\glslongpluralaccessdisplay</code> .....	161
226, 228–231, 233–239, 242, 245, 247,		<code>\glslongtok</code> .....	113, 196, 198, 218–222, 224, 226, 228,
249–252, 254, 256, 259, 261, 263–271,		231, 233–239, 241, 242, 245, 246, 249,	
273, 275, 277, 279–283, 285, 286, 288–		250, 252, 254, 256, 259, 261, 263–271,	
293, 295–298, 300, 302–305, 307, 308, 310		273, 275, 278, 280, 282, 285, 286, 288–	
<code>\glsletentryfield</code> .....	182	293, 295–298, 300, 301, 303–305, 308–310	
<code>\glslink</code> .....	114	<code>\glslonguserfont</code> .....	285, 287–290, 292
<code>\glslink options</code>		<code>\glsmcols</code> .....	397–401
counter .....	10	<code>\GLSname</code> .....	316
format .....	181	<code>\Glsname</code> .....	316
hyper .....	310	<code>\glsname</code> .....	316
hyperoutside .....	65	<code>\glsnameaccessdisplay</code> ..	154, 175, 176, 178
noindex .....	8, 9, 83, 310	<code>\glsnamefont</code> .....	175, 177, 178, 180
textformat .....	67	<code>\glsnavhyperlink</code> .....	126
theHvalue .....	66	<code>\glsnavhyperlinkname</code> .....	87
thevalue .....	66, 145	<code>\glsnavhypertarget</code> .....	369, 370, 380, 382, 384, 396–401
wrgloss .....	8, 64	<code>\glsnavigation</code> .....	369, 370, 380, 382, 384, 396–401
<code>\glslinkcheckfirsthyperhook</code> .....	83	<code>\glsnextpages</code> .....	120
<code>\glslinkpostsetkeys</code> .....	10, 66, 148	<code>\glsnoidxdisplayloc</code> .....	123, 124
<code>\glslinkpresetkeys</code> .....	10, 66, 148	<code>\glsnoidxdisplayloclisthandler</code> ....	123
<code>\glslinkvar</code> .....	86	<code>\glsnoidxloclist</code> .....	124, 144
<code>\glslistchildpostlocation</code> .....	368	<code>\glsnoidxnumberlistloophandler</code> ....	124
<code>\glslistchildprelocation</code> .....	368, 369	<code>\glsnonextpages</code> .....	120
<code>\glslistdesc</code> .....	368, 369	<code>\glsnonumberlistfalse</code> .....	59
<code>\glslistdottedwidth</code> .....	367	<code>\glsnonumberlisttrue</code> .....	59
<code>\glslistgroupheaderfmt</code> .....	369, 370	<code>\glsnopostdotfalse</code> .....	121
<code>\glslistnavigationitem</code> .....	369, 370	<code>\glsnopostdottrue</code> .....	121
<code>\glslistprelocation</code> .....	368, 369	<code>\glsnumberlistloop</code> .....	118
<code>\glslocalunset</code> .....	63, 148	<code>\glsnumlistlastsep</code> .....	123
<code>\glslongaccessdisplay</code> .....	160	<code>\glsnumlistsep</code> .....	123
<code>\glslongdefaultfont</code> ....	200, 218, 220,	<code>\glsopenbrace</code> .....	47, 132, 133
222, 226, 228, 231, 235, 237, 238, 240–		<code>\glsorder</code> .....	116
244, 249, 251, 252, 254, 256–258, 263,		<code>\glspagelistwidth</code> .....	371, 373, 375, 377
267, 270, 271, 273, 274, 277, 285, 294, 307		<code>\glspar</code> .....	142
<code>\glslongemfont</code> ..	263, 265, 268, 275, 278, 279	<code>\GLSpl</code> .....	101, 112, 147
<code>\glslongfont</code> .....	91, 92, 200, 206, 207,	<code>\Glspl</code> .....	56, 101, 112, 147
210, 211, 218–221, 223, 225, 226, 228,			
231–233, 235, 237, 238, 240, 242, 243,			
245, 247, 249, 251, 252, 254, 256, 257,			
259, 261, 263, 265, 267, 268, 270, 271,			
273, 275, 277, 279, 281, 283, 285, 287,			
289, 292, 295, 297, 300, 303, 305, 308, 309			

<code>\glsp1</code> .....	56, 101, 112, 147	<code>\glstextaccessdisplay</code> .....	154, 155
<code>\GLSplural</code> .....	317, 318	<code>\glstextformat</code> .....	64, 67
<code>\Glsplural</code> .....	318	<code>\glstextup</code> .....	234
<code>\glsplural</code> .....	317	<code>\glstreechilddesc</code> .....	379, 381
<code>\glspluralaccessdisplay</code> .....	155	<code>\glstreechildpredesc</code> .....	381
<code>\glspluralsuffix</code> .....	136, 196, 200	<code>\glstreechildprelocation</code> ...	379, 381, 383
<code>\glspostdescription</code> .....	16, 17, 121, 188, 367, 368, 370–377, 380–382, 384	<code>\glstreechildsymbol</code> .....	379, 381
<code>\glspostinline</code> .....	378	<code>\glstreedefaultnamefmt</code> .....	378, 379
<code>\glspostlinkhook</code> .	62, 64, 77–81, 93, 201–211	<code>\glstreedesc</code> .....	379–381
<code>\glsprestandardsort</code> .....	118	<code>\glstreegroupheaderfmt</code> .....	380, 382–384, 395–401, 403
<code>\glsresetentrylist</code> .....	126, 140	<code>\glstreeindent</code> .....	381, 383, 393–395
<code>\glsee</code> .....	48–50	<code>\glstreeitem</code> .....	379, 397, 405
<code>\glseeformat</code> .....	46, 52, 117, 123, 124	<code>\glstreenamebox</code> .....	394, 395
<code>\glseeelist</code> .....	47	<code>\glstreenamefmt</code> .....	378, 379, 381, 383, 385, 387–395
<code>\glsssetabbrvfmt</code> .....	61, 69, 90, 91, 173–178, 180, 186, 200–205, 208, 209, 211	<code>\glstreenavigationfmt</code>	380, 382, 384, 396–401
<code>\glsssetAttribute</code> .....	218– 224, 226, 228–231, 233–236, 238, 239, 242, 245, 247, 249–252, 254, 256, 259, 261, 263–266, 268–271, 273, 275, 277, 279–281, 283, 285, 286, 288, 289, 291– 293, 295–298, 300, 302–305, 307, 308, 310	<code>\glstreenonamechilddesc</code> .....	383
<code>\glsssetcategoryattribute</code> .....	101, 113, 115, 147, 153, 168, 169, 171, 172, 187	<code>\glstreenonamedesc</code> .....	382, 383
<code>\glsssetnoexpandfield</code> .....	12, 13	<code>\glstreenonamesymbol</code> .....	383
<code>\glsssettoctitle</code> .....	120	<code>\glstreepredesc</code> .....	380, 382
<code>\glssshortaccessdisplay</code> .....	159	<code>\glstreeprelocation</code> ...	379, 381, 383, 384
<code>\glssshortpltok</code> .....	197, 198, 218–222, 224, 226, 228, 235–239, 245, 246, 249–252, 254, 259, 261, 263–271, 280–282, 285, 286, 288– 293, 295, 296, 300, 302–305, 307, 308, 310	<code>\glstreesubitem</code> .....	379, 405
<code>\glssshortpluralaccessdisplay</code> .....	160	<code>\glstreesubsubitem</code> .....	379, 406
<code>\glssshorttok</code> .....	113, 196–198, 217–222, 224, 226, 228, 233, 234, 236–239, 241, 245, 246, 249, 250, 252, 254, 256, 259, 261, 263–271, 273, 275, 280, 282, 285, 286, 288, 290–293, 295, 296, 298, 300, 302–305, 307, 308, 310	<code>\glstreesymbol</code> .....	379, 381
<code>\glsssubentryitem</code> .....	138, 367–377, 379, 381, 383, 395, 405	<code>\GLstrLetField</code> .....	34
<code>\glsssymbolaccessdisplay</code> .....	157	<code>\glstype</code> .....	63, 66, 77–81, 148, 201–211
<code>\glsssymbolpluralaccessdisplay</code> .	157, 158	<code>\glunset</code> .....	51, 63, 104, 105, 148
<code>\glstarget</code> .....	138, 139, 367–377, 379, 381, 383, 394, 395, 405, 406	<code>\gluserdescription</code> ...	285, 286, 289, 292
<code>\GLStext</code> .....	316, 317	<code>\glswrite</code> .....	47, 116
<code>\Glstext</code> .....	317	<code>\glswriteentry</code> .....	8, 9
<code>\glstext</code> .....	316, 317	<code>\Glsxtr</code> .....	56
		<code>\glxtr</code> .....	56
		<code>\glxtr@@do@wrglossary</code> .....	8, 10, 12, 13
		<code>\glxtr@addloclistfield</code> .....	14
		<code>\glxtr@addunused</code> .....	50, 51
		<code>\glxtr@applyabbrvfmt</code> .....	211
		<code>\glxtr@applyabbrvstyle</code> ...	194, 196, 215
		<code>\glxtr@counterrecord</code> .....	137
		<code>\glxtr@do@alsoindex@wrglossary</code> ...	14
		<code>\glxtr@doooption</code> .....	5, 16, 17, 23, 24, 26
		<code>\glxtr@fields</code> .....	135
		<code>\glxtr@headentry@p</code> .....	94, 95
		<code>\glxtr@hyperoutsidefalse</code> .....	65
		<code>\glxtr@hyperoutsidettrue</code> .....	65
		<code>\glxtr@ifnextpunc</code> .....	192
		<code>\glxtr@ifpunctoken</code> .....	192
		<code>\glxtr@inc@linkcount</code> .....	66, 153
		<code>\glxtr@inc@wrglossaryctr</code> ...	8, 9, 23, 27

<code>\glxtr@indexonly@saveentrycounter</code>	<code>\glxtrAltTreeSetHangIndent</code> ...	384, 394
..... 13, 14, 27	<code>\glxtrAltTreeSetSubHangIndent</code> ...	395
<code>\glxtr@keylist</code> .....	<code>\glxtralttreeSubSymbolDescLocation</code>	395
<code>\glxtr@label</code> .....	<code>\glxtralttreeSymbolDescLocation</code> ..	
<code>\glxtr@langtag</code> .....	..... 384, 395	
<code>\glxtr@linkprefix</code> .....	<code>\glxtrassignfieldfont</code> .....	69–76
135, 136	<code>\glxtrautoindex</code> .....	182
<code>\glxtr@loaddialect</code> .....	<code>\glxtrautoindexassignsort</code> .....	182
330, 365	<code>\glxtrautoindexentry</code> .....	182
<code>\glxtr@makeglossaries</code> .....	<code>\glxtrbibaddress</code> .....	332
116	<code>\glxtrbibauthor</code> .....	332
<code>\glxtr@newabbreviation</code> .....	<code>\glxtrbibbooktitle</code> .....	332
114, 195	<code>\glxtrbibchapter</code> .....	332
<code>\glxtr@next</code> .....	<code>\glxtrbibedition</code> .....	332
193	<code>\glxtrbibhowpublished</code> .....	332
<code>\glxtr@org@do@wrglossary</code> .....	<code>\glxtrbibinstitution</code> .....	332
27	<code>\glxtrbibjournal</code> .....	332
<code>\glxtr@org@dohyperlink</code> .....	<code>\glxtrbibmonth</code> .....	332
87	<code>\glxtrbibnote</code> .....	332
<code>\glxtr@org@getgrouptitle</code> .....	<code>\glxtrbibnumber</code> .....	332
125	<code>\glxtrbiborganization</code> .....	332
<code>\glxtr@org@newignoredglossary</code> ....	<code>\glxtrbibpages</code> .....	332
40	<code>\glxtrbibpublisher</code> .....	332
<code>\glxtr@org@makenoidxglossaries</code> ....	<code>\glxtrbibschool</code> .....	332
52	<code>\glxtrbibseries</code> .....	332
<code>\glxtr@pluralsuffixes</code> .....	<code>\glxtrbibtitle</code> .....	332
135, 136	<code>\glxtrbibtype</code> .....	333
<code>\glxtr@process</code> .....	<code>\glxtrbibvolume</code> .....	333
140, 141	<code>\glxtrbookindexatendgroup</code> .....	405
<code>\glxtr@provideignoredglossary</code> ....	<code>\glxtrbookindexatsubendgroup</code> ....	405
42	<code>\glxtrbookindexatsubsubendgroup</code> ..	405
<code>\glxtr@punclist</code> .....	<code>\glxtrbookindexbetween</code> .....	405
192, 193	<code>\glxtrbookindexbookmark</code> .....	406
<code>\glxtr@record</code> .....	<code>\glxtrbookindexcols</code> .....	404
11, 135	<code>\glxtrbookindexcolspread</code> .....	404
<code>\glxtr@record@nr</code> .....	<code>\glxtrbookindexfirstmarkfmt</code> ....	407
13	<code>\glxtrbookindexformatheader</code> ....	406
<code>\glxtr@recordsee</code> .....	<code>\glxtrbookindexgroupskip</code> .....	406
12	<code>\glxtrbookindexlastmarkfmt</code> .....	407
<code>\glxtr@resource</code> .....	<code>\glxtrbookindexmulticolenv</code> ....	404
134, 135	<code>\glxtrbookindexname</code> .....	402, 405
<code>\glxtr@s@newignoredglossary</code> ....	<code>\glxtrbookindexparentchildsep</code>	402, 404
40	<code>\glxtrbookindexparentschildsep</code> ..	
<code>\glxtr@s@provideignoredglossary</code> ...	..... 404, 405	
42	<code>\glxtrbookindexprelocation</code> ...	402, 405
<code>\glxtr@saveentrycounter</code> .... 8, 9, 12, 84	<code>\glxtrbookindexsubbetween</code> .....	405
<code>\glxtr@setaccessdisplay</code> .....	<code>\glxtrbookindexsubname</code> .....	406
180	<code>\glxtrbookindexsubprelocation</code> ....	406
<code>\glxtr@setbookindexmark</code> .....	<code>\glxtrbookindexsubsubbetween</code> ....	405
407		
<code>\glxtr@setup@record</code> .....		
13, 14, 26, 27		
<code>\glxtr@shortcutsval</code> .....		
135, 136		
<code>\glxtr@texencoding</code> .....		
136		
<code>\glxtr@undefaction@nr</code> .....		
7		
<code>\glxtr@undefaction@val</code> .....		
7		
<code>\glxtr@usesee</code> .....		
45		
<code>\glxtr@warnonexistsordo</code> . 7, 13, 14, 44, 45		
<code>\glxtr@writefields</code> .....		
134		
<code>\glxtrabbreviationfont</code> .....		
61		
<code>\glxtrabbrvfootnote</code> .....		
..... 222–224, 245–247, 259–261, 280–282		
<code>\glxtrabbrvpluralsuffix</code> .....		
136,		
200, 218, 220, 223, 225, 226, 228, 231,		
234, 249, 262, 285, 294, 297, 300, 305, 308		
<code>\glxtrabbrvtype</code> .....		
17, 18, 198		
<code>\glxtractivatenopost</code> .....		
120		
<code>\glxtraddallcrossrefs</code> .....		
50		
<code>\glxtralias</code> .....		
84		
<code>\glxtrAltTreeIndent</code> .....		
384, 385		
<code>\glxtralttreeInit</code> .....		
394, 400, 401		
<code>\glxtrAltTreePar</code> .....		
384		

<code>\glxtrbookindexthepage</code> .....	407	<code>\glxtrfull</code> .....	18, 19, 321, 322
<code>\glxtrcat</code> .....	55, 56	<code>\Glsxtrfullformat</code> .....	
<code>\glxtrchecknohyperfirst</code> .....	70–72	. 199, 213, 215, 216, 218, 221, 223, 225,	
<code>\glxtrcombingdiacriticIIrules</code> .	336	227, 229, 232, 235, 237, 239, 240, 243,	
<code>\glxtrcombingdiacriticIIrules</code> ..	336	244, 246, 247, 249, 251, 253, 255, 257,	
<code>\glxtrcombingdiacriticIrules</code> ...	336	258, 260, 262, 263, 265, 267, 269, 271,	
<code>\glxtrcombingdiacriticIVrules</code> ..	336	272, 274, 276, 278, 280, 281, 283, 286,	
<code>\glxtrComputeTreeIndent</code> .....	394, 395	287, 289, 292, 295, 298, 301, 303, 306, 308	
<code>\glxtrComputeTreeSubIndent</code> .....	395	<code>\glxtrfullformat</code> .....	
<code>\glxtrcounterprefix</code> .....	128	..... 199, 213, 215, 216, 218, 220,	
<code>\glxtrcurrencyrules</code> .....	339	223, 225, 227, 229, 232, 235, 237, 239,	
<code>\Glsxtrdefaultsubsequentfmt</code> ...	214, 215	240, 243–245, 247, 249, 251, 253, 254,	
<code>\glxtrdefaultsubsequentfmt</code> ...	213, 215	257–259, 261, 263, 265, 267, 269, 271,	
<code>\Glsxtrdefaultsubsequentplfmt</code> .	214, 215	272, 274, 276, 278, 280, 281, 283, 286,	
<code>\glxtrdefaultsubsequentplfmt</code> .	214, 215	287, 289, 292, 295, 298, 301, 303, 306, 308	
<code>\GlsXtrDefineAbbreviationShortcuts</code> .	20	<code>\GLSxtrfullpl</code> .....	18, 19, 322, 323
<code>\GlsXtrDefineAcShortcuts</code> .....	20, 21	<code>\Glsxtrfullpl</code> .....	18, 19, 323
<code>\GlsXtrDefineOtherShortcuts</code> .....	20, 21	<code>\glxtrfullpl</code> .....	18, 19, 322
<code>\glxtrdetoklocation</code> .....	147	<code>\Glsxtrfullplformat</code> .....	
<code>\glxtrdiscardperiod</code> .....	189	. 199, 213, 215, 216, 218, 221, 223, 225,	
<code>\glxtrdisplayendloc</code> .....	127	227, 229, 232, 235, 237, 239, 240, 243,	
<code>\glxtrdisplayendloohook</code> .....	128	244, 246, 247, 250, 251, 253, 255, 257,	
<code>\glxtrdisplaysingleloc</code> .....	127, 128	259, 260, 262, 264, 265, 267, 269, 271,	
<code>\glxtrdisplaystartloc</code> .....	127	272, 275, 276, 278, 280, 281, 283, 286,	
<code>\glxtrdoautoindexname</code> .....	85, 86, 179	287, 289, 292, 295, 298, 301, 303, 306, 309	
<code>\glxtrdopostpunc</code> .....	224, 247, 261, 282	<code>\glxtrfullplformat</code> .....	
<code>\glxtrdowrglossaryhook</code> .....	85	.... 212, 213, 215, 216, 218, 220, 223,	
<code>\GlsXtrDualField</code> .....	332	225, 227, 229, 232, 235, 237, 239, 240,	
<code>\glxtrremsuffix</code> .....	263, 265, 267,	243–245, 247, 249, 251, 253, 254, 257,	
268, 270, 271, 273, 275, 277, 279, 281, 283		258, 260, 261, 263, 265, 267, 269, 271,	
<code>\GlsXtrEnableEntryCounting</code> .....	112	272, 274, 276, 278, 280, 281, 283, 286,	
<code>\GlsXtrEnableEntryUnitCounting</code> ....	101	287, 289, 292, 295, 298, 301, 303, 306, 308	
<code>\GlsXtrEnableOnTheFly</code> .....	54, 57	<code>\glxtrfullsep</code> .....	198, 199,
<code>\glxtrendfor</code> .....	32	218–221, 223–230, 232, 234–242, 244,	
<code>\glxtrfieldlistgadd</code> .....	137	246, 248–258, 260, 262–270, 272–274,	
<code>\glxtrfieldtitlecase</code> .....	174–177, 180	276–279, 282–284, 294–299, 302–305, 309	
<code>\glxtrfieldtitlecasecs</code> .....	173	<code>\glxtrngenabbrvfmt</code> .....	61
<code>\glxtrfieldxifinlist</code> .....	142	<code>\glxtrgeneralpuncIIrules</code> .....	339
<code>\glxtrfirstscfont</code> .....	234	<code>\glxtrgeneralpuncIrules</code> .....	339
<code>\glxtrfirstsmfont</code> .....	248	<code>\glxtrgetgroupitle</code> .....	126, 406
<code>\GlsXtrFmtDefaultOptions</code> .....	30	<code>\glxtrgroupfield</code> .....	143
<code>\glxtrfmtdisplay</code> .....	30	<code>\Glsxtrheadfirst</code> .....	313
<code>\GlsXtrFmtField</code> .....	30, 31	<code>\glxtrheadfirst</code> .....	313
<code>\glxtrfootnotename</code> .....		<code>\Glsxtrheadfirstplural</code> .....	313
.... 222, 224, 245, 246, 259, 261, 280, 282		<code>\glxtrheadfirstplural</code> .....	313
<code>\GlsXtrForeignTextField</code> .....	36	<code>\Glsxtrheadfull</code> .....	313
<code>\GlsXtrFormatLocationList</code> 59, 61, 391–393		<code>\glxtrheadfull</code> .....	313
<code>\GLSxtrfull</code> .....	18, 19, 321, 322	<code>\Glsxtrheadfullpl</code> .....	313
<code>\Glsxtrfull</code> .....	18, 19, 322	<code>\glxtrheadfullpl</code> .....	313

<code>\Glsxtrheadlong</code> .....	313	244, 246, 248, 252, 254–256, 258, 260,
<code>\glxtrheadlong</code> .....	313	262, 270, 271, 273, 274, 276, 277, 279,
<code>\Glsxtrheadlongpl</code> .....	313	281, 283, 287, 290, 297, 301, 306, 309, 328
<code>\glxtrheadlongpl</code> .....	313	<code>\Glsxtrinilinefullplformat</code> 199, 203, 215,
<code>\Glsxtrheadname</code> .....	313	216, 224, 225, 227, 229, 230, 232, 239–
<code>\glxtrheadname</code> .....	137, 138, 313	242, 244, 246, 248, 253–255, 257, 258,
<code>\Glsxtrheadplural</code> .....	313	260, 262, 270, 272–274, 276, 278, 279,
<code>\glxtrheadplural</code> .....	313	282, 284, 287, 290, 298, 301, 306, 309, 329
<code>\Glsxtrheadshort</code> .....	313	<code>\glxtrinilinefullplformat</code> .....
<code>\glxtrheadshort</code> .....	313	..... 199, 202, 203, 215,
<code>\Glsxtrheadshorttpl</code> .....	313	216, 223, 225, 226, 228, 230, 232, 238,
<code>\glxtrheadshorttpl</code> .....	313	240–242, 244, 246, 248, 253–256, 258,
<code>\Glsxtrheadtext</code> .....	313	260, 262, 270, 272–274, 276, 277, 279,
<code>\glxtrheadtext</code> .....	313	282, 283, 287, 290, 297, 301, 306, 309, 329
<code>\glxtrhyperlink</code> .....	23, 24, 88	<code>\glxtrininsertinsidefalse</code> .....
<code>\glxtrhyphensuffix</code> .....	295, 303	..... 217
<code>\glxtrifcounttrigger</code> .....	104, 105	<code>\GlsXtrInternalLocationHyperlink</code> 24, 128
<code>\glxtrifcustomdiscardperiod</code> .....	189	<code>\glxtrLatinA</code> .....
<code>\glxtrifemptyglossary</code> .....	126, 133, 140	..... 341–346
<code>\GlsXtrIfFieldCmpNum</code> .....	33	<code>\glxtrLatinAELigature</code> .....
<code>\GlsXtrIfFieldEqNum</code> .....	138	..... 343, 345, 346
<code>\GlsXtrIfFieldNonZero</code> .....	331	<code>\glxtrLatinE</code> .....
<code>\glxtrifhasfield</code> ..	35, 36, 84, 331, 332, 402	..... 341–346
<code>\glxtrifhyphenstart</code> .....	.....	<code>\glxtrLatinEszettSs</code> .....
.....	294, 296, 299, 302, 304, 305	..... 342–344, 346
<code>\glxtrifindexing</code> .....	85	<code>\glxtrLatinEszettSz</code> .....
<code>\glxtrifinmark</code> .....	68, 94–97, 312, 313	..... 343, 345
<code>\glxtrifnextpunc</code> .....	192, 193	<code>\glxtrLatinEth</code> .....
<code>\glxtrifperiod</code> .....	189, 191, 192	..... 342–345
<code>\glxtrifrecordtrigger</code> .....	149–151	<code>\glxtrLatinH</code> .....
<code>\glxtrifwasfirstuse</code> .....	.....	..... 341–346
.....	69–72, 77–80, 83, 115, 190,	<code>\glxtrLatinI</code> .....
.....	191, 201, 204–211, 224, 225, 247, 261,	..... 341–346
.....	282, 283, 286, 288–290, 300, 302, 305, 307	<code>\glxtrLatinInsularG</code> .....
<code>\glxtrinclinkcounter</code> .....	153	..... 345
<code>\glxtrindexaliased</code> .....	84	<code>\glxtrLatinK</code> .....
<code>\glxtrindexseealso</code> .....	49, 50	..... 341–346
<code>\glxtrinithyperoutside</code> .....	66	<code>\glxtrLatinL</code> .....
<code>\glxtrinitwrgloss</code> .....	66, 148	..... 341–346
<code>\glxtrinitwrglossbeforefalse</code> .....	64	<code>\glxtrLatinM</code> .....
<code>\glxtrinitwrglossbeforetrue</code> .....	64	..... 341–346
<code>\Glsxtrinilinefullformat</code> 199, 201, 215,	.....	<code>\glxtrLatinN</code> .....
.....	216, 224, 225, 227, 229, 230, 232, 239–	..... 341–346
.....	242, 244, 246, 248, 253–255, 257, 258,	<code>\glxtrLatinO</code> .....
.....	260, 262, 270, 272–274, 276, 278, 279,	..... 341–346
.....	282, 283, 287, 290, 298, 301, 306, 309, 329	<code>\glxtrLatinOELigature</code> .....
<code>\glxtrinilinefullformat</code> .....	.....	..... 343, 345, 346
.....	199, 201, 202, 215, 216, 223,	<code>\glxtrLatinP</code> .....
.....	225, 226, 228, 230, 232, 238, 240–242,	..... 341–346
.....	.....	<code>\glxtrLatinS</code> .....
.....	.....	..... 341–346
.....	.....	<code>\glxtrLatinT</code> .....
.....	.....	..... 341–346
.....	.....	<code>\glxtrLatinThorn</code> .....
.....	.....	..... 345
.....	.....	<code>\glxtrLatinX</code> .....
.....	.....	..... 341–346
.....	.....	<code>\glxtrlocationhyperlink</code> .....
.....	.....	..... 128
.....	.....	<code>\glxtrlocrangefmt</code> .....
.....	.....	..... 127, 128
.....	.....	<code>\GLSxtrlong</code> .....
.....	.....	..... 18, 19, 320
.....	.....	<code>\Glsxtrlong</code> .....
.....	.....	..... 18, 19, 321
.....	.....	<code>\glxtrlong</code> .....
.....	.....	..... 18, 19, 320
.....	.....	<code>\glxtrlonghyphen</code> .....
.....	.....	..... 301
.....	.....	<code>\glxtrlonghyphennoshort</code> .....
.....	.....	..... 297, 298
.....	.....	<code>\glxtrlonghyphenshort</code> .....
.....	.....	..... 295
.....	.....	<code>\glxtrlongnoshortdescname</code> .
.....	.....	..... 231, 278, 297
.....	.....	<code>\glxtrlongnoshortname</code> .....
.....	.....	.....
.....	.....	..... 233, 241, 256, 273, 275, 298
.....	.....	<code>\GLSxtrlongpl</code> .....
.....	.....	..... 18, 19, 320, 321
.....	.....	<code>\Glsxtrlongpl</code> .....
.....	.....	..... 18, 19, 321



<code>\glxstrlongpl</code> .....	18, 19, 320	<code>\glxstronlydescsort</code> .....	309
<code>\glxstrlongshortdescname</code> .....		<code>\glxstronlyname</code> .....	308
.....	219, 236, 250, 264, 266, 296, 301	<code>\glxstronlysuffix</code> .....	308
<code>\glxstrlongshortdescsort</code> .....		<code>\glxstrorg@ifKV@glslink@hyper</code> .....	62
.....	219, 236, 250, 264, 266, 291, 296, 301	<code>\glxstrorglong</code> .....	196, 219, 297
<code>\glxstrlongshortname</code> .....		<code>\glxstrorgshort</code> .....	196, 219
.....	218, 234, 249, 263, 264, 285, 286, 295, 300	<code>\GLSxtrp</code> .....	94
<code>\glxstrlongshortuserdescname</code> ..	288, 291	<code>\Glsxtrp</code> .....	94
<code>\glxstrmarkhook</code> .....	311	<code>\glxstrp</code> .....	93, 95
<code>\glxstrMathItalicAlpha</code> .	350, 351, 354, 355	<code>\glxstrparen</code> .....	190,
<code>\glxstrMathItalicBeta</code> ..	350, 351, 354, 355	.....	191, 198, 199, 218–221, 223–230, 232,
<code>\glxstrMathItalicChi</code> .....	351, 355, 356	.....	234–244, 246, 248–258, 260, 262–274,
<code>\glxstrMathItalicDelta</code> .	350, 351, 354, 355	.....	276–279, 282–284, 294–299, 302–305, 309
<code>\glxstrMathItalicEpsilon</code> ..	350, 351, 354, 355	<code>\Glsxtrpl</code> .....	56
<code>\glxstrMathItalicEta</code> ...	350, 351, 354, 355	<code>\glxstrpl</code> .....	56
<code>\glxstrMathItalicGamma</code> .	350, 351, 354, 355	<code>\glxstrpostdescription</code> .	121, 171, 188, 378
<code>\glxstrMathItalicIota</code> ..	350, 351, 354, 355	<code>\glxstrposthyphenlong</code> .....	305, 307
<code>\glxstrMathItalicKappa</code> .	350, 351, 354, 355	<code>\glxstrposthyphenshort</code> .....	300, 302
<code>\glxstrMathItalicLambda</code> ..	350, 351, 354, 355	<code>\glxstrposthyphensequent</code> .....	
<code>\glxstrMathItalicMu</code> ....	350, 351, 354, 355	.....	300, 302, 305, 307
<code>\glxstrMathItalicNu</code> ....	350, 351, 354, 356	<code>\glxstrpostlink</code> .....	189
<code>\glxstrMathItalicOmega</code> .....	351, 355, 356	<code>\glxstrpostlinkendsentence</code> .....	189
<code>\glxstrMathItalicOmicron</code> ..	350, 351, 355, 356	<code>\glxstrpostlinkhook</code> .....	189
<code>\glxstrMathItalicPhi</code> .....	351, 355, 356	<code>\glxstrpostlocalreset</code> .....	100, 102, 110
<code>\glxstrMathItalicPi</code> ....	350, 351, 355, 356	<code>\glxstrpostlocalunset</code> .....	99, 102, 110
<code>\glxstrMathItalicPsi</code> .....	351, 355, 356	<code>\glxstrpostlongdescription</code> .....	40
<code>\glxstrMathItalicRho</code> ...	350, 351, 355, 356	<code>\glxstrpostnamehook</code> .....	176–178, 181
<code>\glxstrMathItalicSigma</code> .....	351, 355, 356	<code>\GlsXtrPostNewAbbreviation</code> .....	198,
<code>\glxstrMathItalicTau</code> .....	351, 355, 356	.....	215, 216, 218–222, 224, 226–231, 233–
<code>\glxstrMathItalicTheta</code> .	350, 351, 354, 355	.....	236, 238, 239, 242, 245, 247, 249–252,
<code>\glxstrMathItalicUpsilon</code> ...	351, 355, 356	.....	254, 256, 259, 261, 263–266, 268–271,
<code>\glxstrMathItalicXi</code> ....	350, 351, 355, 356	.....	273, 275, 277, 279–282, 285, 286, 288–
<code>\glxstrMathItalicZeta</code> ..	350, 351, 354, 355	.....	293, 295–298, 300, 302–305, 307, 308, 310
<code>\glxstrnewabbrevpresetkeyhook</code> .....	197	<code>\glxstrpostreset</code> .....	100, 102, 110
<code>\glxstrnewnumber</code> .....	19	<code>\glxstrpostunset</code> .....	98, 102, 110
<code>\glxstrnewsymbol</code> .....	19	<code>\glxstrprelocation</code> .....	
<code>\glxstrNoGlossaryWarning</code> .....	14, 21, 129	.....	368, 370, 372, 374, 376, 379, 402
<code>\GlsXtrNoGlsWarningAutoMake</code> .....	133	<code>\glxstrprotectlinks</code> .....	87–89
<code>\GlsXtrNoGlsWarningBuildInfo</code> .....	133	<code>\GlsXtrRecordCounter</code> .....	11
<code>\GlsXtrNoGlsWarningCheckFile</code> .....	133	<code>\glxstrrecordtriggervalue</code> .....	148
<code>\GlsXtrNoGlsWarningEmptyMain</code> .....	133	<code>\GlsXtrRecordWarning</code> .....	134
<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	133	<code>\glxstrregularfont</code> .....	61, 69
<code>\GlsXtrNoGlsWarningEmptyStart</code> .....	133	<code>\glxstrresourcecount</code> .....	135
<code>\GlsXtrNoGlsWarningHead</code> .....	132	<code>\glxstrresourcefile</code> .....	135
<code>\GlsXtrNoGlsWarningMisMatch</code> .....	133	<code>\glxstrresourceinit</code> .....	134
<code>\GlsXtrNoGlsWarningNoOut</code> .....	133	<code>\glxstrrestoremarkhook</code> .....	311
<code>\GlsXtrNoGlsWarningTail</code> .....	133, 134	<code>\glxstrrestorepostpunc</code> .....	121
<code>\glxstrnopostpunc</code> .....	120	<code>\glxstrscfont</code> .....	234
<code>\glxstronlydescname</code> .....	309		

<code>\glxtrscsuffix</code> .....	<code>\Glsxtrtitlefirstplural</code> ....
.... 235, 237, 238, 240, 242, 243, 245, 247	312–314, 327
<code>\GlsXtrSetActualChar</code> .....	<code>\glxtrtitlefirstplural</code> ....
185	312, 313, 327
<code>\glxtrsetaliasnoindex</code> .....	<code>\Glsxtrtitlefull</code> .....
13, 14, 84	312–314, 329
<code>\GlsXtrSetEncapChar</code> .....	<code>\glxtrtitlefull</code> .....
185	312–314, 328
<code>\GlsXtrSetEscChar</code> .....	<code>\Glsxtrtitlefullpl</code> .....
185	312–314, 329
<code>\glxtrsetfieldifexists</code> .....	<code>\glxtrtitlefullpl</code> .....
34, 35	312–314, 329
<code>\GlsXtrSetLevelChar</code> .....	<code>\Glsxtrtitlelong</code> .....
185	312–314, 327, 328
<code>\glxtrsetpopts</code> .....	<code>\glxtrtitlelong</code> .....
93	312–314, 327
<code>\glxtrsetupfulldefs</code> .....	<code>\Glsxtrtitlelongpl</code> .....
..... 201–203, 225, 247, 261, 283	312–314, 328
<code>\GLSxtrshort</code> .....	<code>\glxtrtitlelongpl</code> .....
18, 19, 97, 314, 315	312–314, 328
<code>\Glsxtrshort</code> .....	<code>\Glsxtrtitlename</code> .....
18, 19, 315	312, 313, 325
<code>\glxtrshort</code> .....	<code>\glxtrtitlename</code> .....
18, 19, 314	312, 313, 324
<code>\glxtrshortdescname</code> ...	<code>\glxtrtitleorpdforheading</code> .....
228, 239, 254, 271	..... 25, 137–139, 312, 313
<code>\glxtrshorthyphen</code> .....	<code>\Glsxtrtitleplural</code> .....
306	312, 313, 326
<code>\glxtrshorthyphenlong</code> .....	<code>\glxtrtitleplural</code> ....
303	312, 313, 325, 326
<code>\glxtrshortlongdescname</code> .....	<code>\Glsxtrtitleshort</code> .....
..... 221, 237, 251, 267, 269, 304, 307	312, 313, 324
<code>\glxtrshortlongdescsort</code> .....	<code>\glxtrtitleshort</code> .....
..... 221, 237, 252, 267, 269, 293, 304, 307	312, 313, 323
<code>\glxtrshortlongname</code> .....	<code>\Glsxtrtitleshortpl</code> .....
220, 236, 250, 266, 268, 288, 292, 303, 305	312, 313, 324
<code>\glxtrshortlonguserdescname</code> ..	<code>\glxtrtitleshortpl</code> ....
290, 293	312, 313, 323, 324
<code>\glxtrshortnolongname</code> .	<code>\Glsxtrtitletext</code> .....
226, 238, 252, 270	312, 313, 325
<code>\GLSxtrshortpl</code> .....	<code>\glxtrtitletext</code> .....
18, 19, 314, 315	312, 313, 325
<code>\Glsxtrshortpl</code> .....	<code>\GlsXtrTotalRecordCount</code> .....
18, 19, 315	147
<code>\glxtrshortpl</code> .....	<code>\glxtrtreetopindent</code> .....
18, 19, 314, 315	385, 393
<code>\glxtrsmfont</code> .....	<code>\glxtrundefaction</code> ..
248	7, 13, 14, 28, 41, 43–45
<code>\glxtrsmsuffix</code> .....	<code>\glxtrundeftag</code> .....
..... 249, 251, 252, 254, 256, 257, 259, 261	27, 123, 124
<code>\GlsXtrStandaloneGlossaryType</code> .	<code>\GlsXtrUnknownDialectWarning</code> .....
138, 139	37
<code>\GlsXtrStandaloneSubEntryItem</code> .	<code>\glxtrunsrtdo</code> .....
138, 139	141, 142
<code>\Glsxtrsubsequentfmt</code> .....	<code>\glxtrUpAlpha</code> .....
..... 212, 215, 231, 242, 244,	349, 350, 354, 355
256, 258, 274, 275, 277, 279, 297, 300, 306	<code>\glxtrUpBeta</code> .....
<code>\glxtrsubsequentfmt</code> .....	349, 350, 354, 355
..... 212, 215, 231, 242, 243,	<code>\glxtrUpChi</code> .....
256, 258, 274, 275, 277, 279, 297, 300, 306	349, 350, 355, 356
<code>\Glsxtrsubsequentplfmt</code> .....	<code>\glxtrUpDelta</code> .....
..... 212, 215, 232, 242, 244,	349, 350, 354, 355
256, 258, 274, 275, 277, 279, 297, 301, 306	<code>\glxtrUpDigamma</code> .....
<code>\glxtrsubsequentplfmt</code> .....	349, 350, 354
..... 212, 215, 231, 242, 243,	<code>\glxtrUpEpsilon</code> .....
256, 258, 274, 275, 277, 279, 297, 300, 306	349, 350, 354, 355
<code>\glxtrsupplocationurl</code> .....	<code>\glxtrUpEta</code> .....
128, 129	349, 350, 354, 355
<code>\glxtrtagfont</code> .....	<code>\glxtrUpGamma</code> .....
188	349, 350, 354, 355
<code>\Glsxtrtitlefirst</code> .....	<code>\glxtrUpIota</code> .....
312, 313, 326	349, 350, 354, 355
<code>\glxtrtitlefirst</code> .....	<code>\glxtrUpKappa</code> .....
312, 313, 326	349, 350, 354, 355
	<code>\glxtrUpLambda</code> .....
	349, 350, 354, 355
	<code>\glxtrUpMu</code> .....
	349, 350, 354, 356
	<code>\glxtrUpNu</code> .....
	349, 350, 354, 356
	<code>\glxtrUpOmega</code> .....
	349, 350, 355, 356
	<code>\glxtrUpOmicron</code> .....
	349, 350, 355, 356
	<code>\glxtrUpPhi</code> .....
	349, 350, 355, 356
	<code>\glxtrUpPi</code> .....
	349, 350, 355, 356
	<code>\glxtrUpPsi</code> .....
	349, 350, 355, 356
	<code>\glxtrUpRho</code> .....
	349, 350, 355, 356
	<code>\glxtrUpSigma</code> .....
	349, 350, 355, 356
	<code>\glxtrUpTau</code> .....
	349, 350, 355, 356



<code>\glxtrUpTheta</code> .....	349, 350, 354, 355	<code>\ifcsundef</code> .....	33, 36–39, 41–43, 53, 57, 59, 89, 102, 107–111, 125, 126, 129, 142, 153, 169, 214, 216, 217, 367, 385, 386, 394, 407
<code>\glxtrUpUpsilon</code> .....	349, 350, 355, 356	<code>\ifcsvoid</code> .....	49, 168
<code>\glxtrUpXi</code> .....	349, 350, 355, 356	<code>\ifdef</code> .....	14, 19, 26, 30, 36, 38, 44, 47, 48, 51, 57, 58, 83, 87, 88, 94–96, 119, 123, 124, 136, 145, 171, 172, 185, 188, 284, 312, 323–329, 331, 364, 365, 367–370, 378–384, 395–401, 403, 406, 407
<code>\glxtrUpZeta</code> .....	349, 350, 354, 355	<code>\ifdefempty</code> ....	7–9, 33, 36, 37, 41–43, 45, 46, 66, 68, 101, 113, 116, 119, 127, 140, 143, 147, 148, 162–164, 187, 195, 211, 404
<code>\GlsXtrUseAbbrStyleFmts</code> .....	219, 222, 228, 230, 231, 233, 234, 236, 238, 241, 250, 252, 255, 264, 266, 268, 270, 272, 277, 280, 288, 291, 293, 296, 297, 299, 302, 304, 307, 310	<code>\ifdefequal</code> .....	35, 52, 133, 143, 179
<code>\GlsXtrUseAbbrStyleSetup</code> .....	227, 229, 230, 233, 234, 241, 243, 255, 257, 272, 276, 277, 280	<code>\ifdefstring</code> ....	6, 23, 24, 35, 182, 187, 403
<code>\glxtruserfield</code> .....	284	<code>\ifdefvoid</code> .....	45, 48–50, 88, 107, 124, 125, 128, 143, 144
<code>\glxtruserparen</code> .....	285–293	<code>\ifdim</code> .....	58, 115, 385–393
<code>\glxtrusersuffix</code> .....	285, 287, 289, 292	<code>\IfFileExists</code> .....	22, 129, 133, 134, 136, 365, 366
<code>\glxtruseseealsoformat</code> .....	46, 47	<code>\ifglossaryexists</code> .....	45
<code>\glxtruseseeeformat</code> .....	46	<code>\ifglshasacronym</code> .....	18, 133
<code>\GlsXtrWarnDeprecatedAbbrStyle</code> .....	194, 216	<code>\ifglshasshortcuts</code> .....	20
<code>\GlsXtrWarning</code> .....	55, 56	<code>\ifglshasautomake</code> .....	119, 133, 136
<code>\glxtrword</code> .....	195	<code>\ifglshasentrycounter</code> .....	38
<code>\glxtrwordsep</code> .....	195, 294, 296, 299, 302, 304, 305	<code>\ifglshasentryexists</code> .....	9, 27, 43, 44, 55, 56, 59, 69, 143, 169, 188, 189
<code>\glxtrwrglossmark</code> .....	24	<code>\ifglshasfield</code> .....	167
<b>H</b>			
<code>\hangindent</code> .....	381, 383, 384, 394–396, 400, 401	<code>\ifglshaslong</code> .....	106, 107, 151, 152
<code>\hbox</code> .....	367	<code>\ifglshasparent</code> ....	138–140, 143, 387–389
<code>\hfill</code> .....	367	<code>\ifglshasshort</code> .....	46, 61, 69
<code>\href</code> .....	87	<code>\ifglshassymbol</code> .....	191, 381, 382, 384
<code>\hsize</code> .....	58	<code>\ifglshasindexonlyfirst</code> .....	85
<code>\hss</code> .....	367	<code>\ifglshasnogroupskip</code> .....	368, 370–378, 380, 381, 383, 395, 403
<code>\hyperlink</code> .....	88	<code>\ifglshasnonumberlist</code> .....	61
<code>\hyperpage</code> .....	181	<code>\ifglshasnopostdot</code> .....	16, 121
<code>\hyperref</code> .....	87, 129, 331	<code>\ifglssanitizesort</code> .....	118
<code>hyperref package</code> .....	89, 181, 310, 323	<code>\ifglssubentrycounter</code> .....	38
<b>I</b>			
<code>\if</code> .....	54	<code>\ifglshasused</code> .....	50, 51, 83, 85, 103, 112, 115, 211, 387, 388, 390, 391, 393
<code>\if@glxtr@autoseeindex</code> .....	26, 45, 48	<code>\ifglshasxindy</code> .....	129, 131
<code>\if@glxtr@format@override</code> .....	182	<code>\ifglxtr@hyperoutside</code> .....	67
<code>\if@glxtr@docdefrestricted</code> .....	52	<code>\ifglxtr@trinitwrglossbefore</code> ...	64, 67, 148
<code>\if@glxtr@indexcrossrefs</code> .....	15, 50	<code>\ifglxtr@insertinside</code> .....	204–211, 213, 214, 218, 220, 221, 223–232, 235, 237–251, 253– 265, 267, 269–284, 286, 287, 289, 290, 292, 294, 296, 299–302, 305, 306, 308, 309
<code>\ifblank</code> .....	29, 55, 56, 116	<code>\ifHy@hyperindex</code> .....	181
<code>\ifbool</code> .....	28, 39		
<code>\ifcase</code> .....	7, 13, 20, 21, 24, 53, 64, 122, 379, 405		
<code>\ifcsdef</code> .....	28, 38–43, 65, 67, 81, 82, 93–97, 107, 119, 125, 126, 138, 142, 145–147, 152, 173– 176, 178–180, 190, 194, 211, 215, 370–377		
<code>\ifcsstring</code> .....	28, 169, 214		

<code>\ifinlist</code> .....	99	175, 177–183, 187, 188, 192–197, 200–
<code>\ifinlistcs</code> .....	32, 53	211, 213–215, 225, 247, 261, 283, 310–
<code>\ifinner</code> .....	25	314, 364, 365, 379, 384, 386, 397, 404–407
<code>\ifKV@glslink@hyper</code> .....	62, 66–68	<code>\letabbreviationstyle</code> .. 224, 226, 227,
<code>\ifKV@glslink@local</code> .....	63, 148	229, 232, 233, 239, 241, 253, 255, 271, 272
<code>\ifKV@glslink@noindex</code> ..	8, 9, 12, 30, 84, 85	<code>\letcs</code> .....
<code>\ifmmode</code> .....	25	28, 32, 33, 45, 46, 50,
<code>\ifnum</code> .....	15, 33, 51, 52, 103,	65, 67, 81, 123–125, 142, 143, 173–181, 407
111, 112, 125, 135, 148, 381, 383, 394, 395		<code>\levelchar</code> .....
<code>\ifstrempy</code> .....	138, 145, 153	185
<code>\ifstrequal</code> .....	17, 22	<code>\listadd</code> .....
<code>\ifthenelse</code> .....	133, 190	107
<code>\IfTrackedDialectHasMapping</code> .....	36	<code>\listbreak</code> .....
<code>\IfTrackedLanguageFileExists</code> .....	330	187
<code>\ifundef</code> ....	23, 32, 33, 36, 116, 187, 188, 364	<code>\listcsadd</code> .....
<code>\ifx</code> .....	8–11, 47,	31
57, 59, 67, 116, 120, 122, 127, 128, 136,		<code>\listcseadd</code> .....
182, 183, 185, 193–195, 197, 293, 294, 401		31, 108
<code>\immediate</code> .....	52, 103, 111, 129, 130, 136	<code>\listcsgadd</code> .....
<code>\index</code> .....	182	31, 53
<code>\indexspace</code> .	368, 380–384, 395–401, 403, 406	<code>\listcsxadd</code> .....
<code>\input</code> .....	330	31, 108
<code>\inputencodingname</code> .....	136	<code>\listxadd</code> .....
<code>\InputIfFileExists</code> .....	51	99
<code>\istfilename</code> .....	116	<code>\loadglsentries</code> .....
<code>\item</code> ....	131, 132, 367–370, 379, 380, 396, 397	53, 130
		<code>\long</code> .....
		40
		<b>M</b>
		<code>\MakeAcronymsAbbreviations</code> .....
		115
		<code>\makeatletter</code> .....
		51, 129, 134, 184
		<code>\makeatother</code> .....
		184
		<code>\makebox</code> .....
		367, 394, 395
		<code>\makefirstuc</code> .....
		188
		<code>makeglossaries</code> .....
		122
		<code>\makeglossaries</code> .....
		116, 130–133, 137
		<code>\makeglossary</code> .....
		116
		<code>makeindex</code> .....
		408
		<code>makeindex</code> .....
		14, 115
		<code>\makenoidxglossaries</code> .....
		131
		<code>\MakeTextUppercase</code> .....
		313
		<code>\MakeUppercase</code> .....
		312, 313
		<code>\marginpar</code> .....
		25
		<code>\markboth</code> .....
		311
		<code>\markright</code> .....
		311
		<code>\mathit</code> .....
		334
		<code>\mathrm</code> .....
		333–335
		<code>\maxdimen</code> .....
		58
		<code>\mbox</code> .....
		369, 370, 394
		<code>\medskip</code> .....
		133, 142
		<code>\MessageBreak</code> .....
		.. 53, 57, 103, 112, 116, 119, 120, 134, 214
		<code>mfistuc package</code> .....
		187
		<code>\mfistucMakeUppercase</code> .....
		70–80, 82, 91, 92, 94, 97,
		106, 114, 152, 154–161, 164–167, 176,
		177, 181, 202, 203, 205, 207, 209, 211–213
		<code>\mfu@checkword@arg</code> .....
		187, 188
		<code>\mfu@checkword@do</code> .....
		188

## N

<code>\NeedsTeXFormat</code> .....	5, 331, 366, 402	<code>seealso</code> .....	16, 45, 46, 48–50, 419
<code>\new@atom@glossaryentry</code> .....	52	<code>short</code> .....	159, 166, 194
<code>\new@glossaryentry</code> .....	53, 119	<code>shortaccess</code> .....	162–164
<code>\new@ifnextchar</code> .....	30, 81, 82, 105, 106, 145, 149–151, 192, 200–210	<code>shortaccessplural</code> .....	162
<code>\newabbr</code> .....	18, 19	<code>shortplural</code> .....	160, 167, 194
<code>\newabbreviation</code> .....	18, 19	<code>symbol</code> .....	157, 165
<code>\newabbreviationhook</code> .....	197	<code>symbolplural</code> .....	157, 158, 165, 166
<code>\newabbreviationstyle</code> .....		<code>text</code> 87, 154, 155, 164, 217, 219, 316, 317, 325	
. 217, 219–222, 224, 226–231, 233–239, 241, 243, 245, 246, 249–253, 255–257, 259, 260, 263, 264, 266–273, 275–278, 280, 282, 285, 286, 288, 290, 291, 293, 295, 297, 298, 300, 301, 303, 305, 307–309		<code>textaccess</code> .....	163
<code>\newacronym</code> .....	113, 114	user2 .....	37
<code>\newacronymhook</code> .....	113	<code>\newglossarystyle</code> .....	404
<code>\newacronymstyle</code> .....	114, 115	<code>\newif</code> .....	64, 181, 217
<code>\newcommand</code> .....	5–13, 15–34, 36–46, 49–51, 54–57, 59–62, 64, 65, 68–70, 81, 82, 84–86, 88, 89, 92–103, 105–115, 119–125, 127–132, 134–143, 145–162, 164–173, 179–195, 198–211, 213–217, 219–222, 226, 228, 231, 233, 234, 248, 249, 262, 263, 284, 285, 288, 290, 294, 296, 299, 302, 304, 305, 307– 309, 311–332, 335–364, 366, 368, 378– 382, 384–386, 393, 394, 402, 403, 406, 407	<code>\newlength</code> .....	384, 385
<code>\newcount</code> .....	134, 144	<code>\newnum</code> .....	19
<code>\newcounter</code> .....	23, 152	<code>\newrobustcmd</code> .....	
<code>\newentry</code> .....	19	... 29, 31, 32, 34, 35, 47, 48, 52, 65, 69, 81, 82, 93, 94, 105, 106, 121, 125, 137, 139, 142, 145, 146, 149–151, 180, 187, 188, 200–211, 293, 312, 314–323, 386–393	
<code>\newglossary</code> .....	17, 116	<code>\newsym</code> .....	19
<code>\newglossaryentry</code> .....		<code>\newterm</code> .....	171
... 19, 52, 53, 101, 109, 113, 171, 172, 198		<code>\newtoks</code> .....	194
<code>\newglossaryentry options</code>		<code>\newwrite</code> .....	52, 116
access .....	163	<code>\nobreak</code> ....	369, 370, 380, 382–384, 396–399
alias .....	16, 45, 48–50	<code>\NoCaseChange</code> .....	94–97, 138, 314–323
desc .....	158, 166	<code>\noexpand</code> ..	10, 11, 22, 47, 49–51, 113, 129, 130, 134, 141–143, 152, 183, 198, 366, 405
descplural .....	158, 159, 166	<code>\nofiles</code> .....	132
first 87, 155, 156, 165, 217, 318, 319, 326, 408		<code>\noindent</code> .....	133, 382–384, 397–399, 401
firstaccess .....	164	<code>\nopagebreak</code> ....	380, 382–384, 395–402, 406
firstplural .....	156, 165, 217, 319, 326, 408	<code>\nopostdesc</code> .....	40, 55, 56, 120, 171
group .....	143	<code>\ns@GLSxtrfull</code> .....	201
loclist .....	31	<code>\ns@Glsxtrfull</code> .....	201
long .....	160, 167, 327	<code>\ns@glxtrfull</code> .....	200
longplural .....	161, 167, 328	<code>\ns@GLSxtrfullpl</code> .....	203
name .....	46, 154, 164, 182, 315, 316, 324	<code>\ns@Glsxtrfullpl</code> .....	202
plural .....	155, 164, 165, 217, 317, 318, 325	<code>\ns@glxtrfullpl</code> .....	202
see .....	15, 16, 26, 45, 48, 50, 53, 117	<code>\ns@GLSxtrlong</code> .....	207
		<code>\ns@Glsxtrlong</code> .....	206
		<code>\ns@glxtrlong</code> .....	205, 206
		<code>\ns@GLSxtrlongpl</code> .....	210
		<code>\ns@Glsxtrlongpl</code> .....	210
		<code>\ns@glxtrlongpl</code> .....	209
		<code>\ns@GLSxtrshort</code> .....	205
		<code>\ns@Glsxtrshort</code> .....	204
		<code>\ns@glxtrshort</code> .....	204
		<code>\ns@GLSxtrshortpl</code> .....	209
		<code>\ns@Glsxtrshortpl</code> .....	208
		<code>\ns@glxtrshortpl</code> .....	207

\null ..... 21  
 \number ..... 51, 108–111, 134, 145  
 \numexpr ..... 108, 111

## O

\O ..... 346, 349  
 \o ..... 349  
 \openout ..... 52  
 \or ..... 7, 13, 14, 20, 21, 24, 53, 64, 379, 405  
 \org@glossaryentrynumbers ..... 59, 120  
 \org@glossarytitle ..... 120  
 \org@ifKV@glslink@hyper ..... 66, 68

## P

\p@gl@hyp@opt ..... 86  
 package options:  
 abbreviations ..... 17, 18  
 accsupp ..... 21, 154  
 acronym ..... 17  
 automake ..... 119, 131, 136  
   true ..... 136  
 autoseeindex ..... 26  
   false ..... 26  
 counter  
   wrglossary ..... 23  
 debug  
   showtargets ..... 88  
 docdef ..... 14, 15, 52, 53, 101, 109  
   atom ..... 51  
   false ..... 52, 53  
   restricted ..... 15  
   true ..... 51, 53  
 docdefs  
   restricted ..... 52  
 indexcounter ..... 331  
 nonumberlist ..... 59  
 nopostdot ..... 16  
   false ..... 16  
 numbers ..... 19  
 postdot ..... 16  
 record 7, 13, 52, 62, 116, 134, 200, 202–211, 417  
   alsoindex ..... 8, 10  
   only ..... 8, 132  
 shortcuts ..... 20  
   ac ..... 20  
   all ..... 20  
   false ..... 20  
   none ..... 20  
   true ..... 20

sort  
   use ..... 68  
 style ..... 22  
 stylemods ..... 22  
 symbols ..... 19, 172  
 undefaction ..... 43, 44  
   error ..... 6  
   warn ..... 6  
 xindy ..... 47, 48  
 \PackageError ..... 6, 11, 22, 26, 52,  
   53, 57, 65, 81, 82, 84, 93, 94, 101, 102,  
   109, 111, 112, 114, 116, 118, 119, 126,  
   136, 141, 142, 145, 190, 214–217, 366, 367  
 \PackageWarning ..... 16  
 \PackageWarningNoLine ..... 16  
 \pageref ..... 23, 38  
 \par ..... 132,  
   133, 369, 370, 379, 381–384, 394–401, 403  
 \parindent .. 379, 381, 383–385, 394–401, 404  
 \parskip ..... 379, 381, 383, 397–399, 404  
 \PassOptionsToPackage ..... 5, 22  
 \pdfbookmark ..... 403  
 \preglossarypreamble ..... 39  
 \preto ..... 84  
 \print@noop@unsrtglossaryunit ... 12, 13  
 \print@op@unsrtglossaryunit ..... 14  
 \printabbreviations ..... 17  
 \printglossaries ..... 117, 131  
 \printglossary ..... 17, 117, 131, 132, 134  
 \printglossary options  
   nonumberlist ..... 61  
   type ..... 119  
 \printnoidxglossaries ..... 131  
 \printnoidxglossary ..... 117, 118, 131  
 \printnumbers ..... 19, 172  
 \printsymbols ..... 19, 172  
 \printunsrtglossary ..... 132, 134, 140  
 \printunsrtglossaryentryprocesshook  
   ..... 140, 141  
 \printunsrtglossaryhandler .... 141, 142  
 \printunsrtglossarypredoglossary .. 141  
 \printunsrtglossaryskipentry ..... 140  
 \printunsrtglossaryunit ..... 13, 14, 142  
 \printunsrtglossaryunitsetup ..... 142  
 \ProcessOptions ..... 367  
 \ProcessOptionsX ..... 24  
 \protect ..... 94–97, 164–167, 195, 198,  
   199, 217–224, 226–228, 230–247, 249–

261, 263–271, 273–283, 285, 286, 288– 293, 295–298, 300–305, 307–310, 314–323	\rGLSpl ..... 147
\protected@csedef .... 34, 35, 125, 385, 386	\rGlspl ..... 147
\protected@csxdef ..... 35, 125, 385, 386	\rglspl ..... 147
\protected@edef ..... 35, 57, 65, 87, 113, 124, 125, 142, 143, 182, 198	\rGLSplformat ..... 151
\protected@write ..... . 11, 12, 52, 60, 61, 116, 117, 134–137, 407	\rGlsplformat ..... 150
\providecommand ..... ... 17–19, 29, 47, 61, 82, 84, 103, 111, 116, 129, 130, 135, 331, 333–335, 367, 402	\rglsplformat ..... 149, 152
\ProvidesFile ..... 330	\romannumeral ..... 385, 386, 394
\ProvidesPackage ..... 5, 331, 366, 402	
	<b>S</b>
<b>Q</b>	\s@glstrfmt ..... 30
\quotechar ..... 185	\s@glshyp@opt ..... 86
	\s@glstr@enabletagging ..... 186
<b>R</b>	\s@glstrfmt ..... 29
\raggedright ..... 372, 373, 376, 377, 404	\s@glstrifhasfield ..... 32
\refstepcounter ..... 23	\s@GlsXtrStartUnsetBuffering ..... 98
\relax ..... 7, 13–15, 18– 21, 24, 26, 30, 33, 51–54, 57, 58, 60, 64, 66, 67, 86, 93, 102, 103, 111, 116, 117, 120, 124, 125, 127, 134–136, 143, 148, 183, 185, 187, 190, 194–196, 293, 294, 379, 381, 383, 386–396, 400, 401, 404–407	\s@GlsXtrStopUnsetBuffering ..... 99
relsize package ..... 248	\s@printunstrtglossary ..... 139, 142
\renewcommand 6, 7, 13–18, 20–24, 26, 39–44, 46, 51–54, 56, 57, 59–64, 81, 83, 85, 87– 89, 93, 99–103, 106, 107, 109–122, 124, 126–130, 142, 147, 171, 173–178, 186– 189, 199, 215, 216, 218–293, 295–298, 300–311, 364, 367–384, 394–401, 404–406	\seealsoname ..... 46, 48
\renewenvironment ..... 368, 370– 377, 379, 381, 383, 394, 397–399, 401, 404	\seename ..... 46
\renewglossarystyle ..... ..... 367–377, 379–383, 394–401	\setabbreviationstyle ..... 114, 219, 227
\renewrobustcmd ..... 68, 88	\setacronymstyle ..... 114, 115
\RequireGlossariesExtraLang ... 330, 365	\setentrycounter ..... 127
\RequirePackage ... 5, 14, 21, 22, 25, 366, 402	\SetGenericNewAcronym ..... 115
\reserved@a ..... 192	\setglossarystyle ..... 23, 120, 367, 369, 370, 380, 382, 383, 395–401, 404
\reserved@b ..... 192	\setkeys ..... 9, 22, 26, 30, 66, 68, 85, 113, 120, 148, 196, 197
\reserved@d ..... 193	\setlength ..... . 58, 379, 381, 383, 384, 395, 397–399, 404
\RestoreAcronyms ..... 114, 115	\settowidth ..... 115, 385–394
\rGLS ..... 147	\setupglossaries ..... 5, 26
\rGls ..... 147	\sfcode ..... 16, 17, 190, 378
\rgls ..... 147	\small ..... 25
\rGLSformat ..... 150	soul package ..... 99
\rGlsformat ..... 150	\space ..... 6, 11, 47, 53, 54, 57, 84, 101–103, 109, 111, 112, 114–118, 120, 130, 133, 134, 137, 141, 142, 145, 190, 191, 195, 199, 219, 367, 368, 378, 381, 382, 384, 385, 394, 402
\rglsformat ..... 149, 152	\spacefactor ..... 16, 17, 190, 196, 378
	\stepcounter ..... 153
	\string 6, 11, 12, 47, 52–54, 57, 60, 61, 67, 81, 82, 84, 93, 94, 101–103, 109, 111, 112, 114–120, 129–137, 141, 142, 145, 175, 177, 178, 180, 182, 190, 331, 335–364, 407
	\strut ..... 367–377, 383
	\subglossentry ..... 120, 144, 367–377, 379, 381, 383, 395, 405
	\subitem ..... 379
	\subsubitem ..... 379

## T

<code>\tablehead</code> .....	374–377
<code>\tabletail</code> .....	374–377
<code>\tabularnewline</code> .....	370–378
<code>\TeX</code> .....	131
<code>\texorpdfstring</code> .	30, 31, 94–97, 312, 323–329
<code>\textbf</code> .....	378
<code>textcase</code> package .....	310
<code>\textsc</code> .....	234
<code>\textsmaller</code> .....	248
<code>\texttt</code> .....	25, 130–133
<code>\the</code> .....	113, 114, 128, 135, 185, 198, 217–224, 226, 228–231, 233–239, 241, 242, 245–247, 249–252, 254, 256, 259, 261, 263–271, 273, 275, 277–283, 285, 286, 288–293, 295–298, 300–305, 307–310
<code>\theHglentrycounter</code> ..	8, 10, 11, 66, 68, 148
<code>\theHglentrycounter</code> ....	8–11, 66, 68, 148
<code>\theindex</code> .....	181
<code>\thewrglossary</code> .....	23
<code>\this@dialect</code> .....	330, 365
<code>\thisgrptitle</code> .....	406
<code>\toks@</code> .....	128, 185
<code>\TrackedDialectClosestSubMatch</code> ....	36
<code>tracklang</code> package .....	36, 136, 335
<code>\TrackLangGetDefaultScript</code> .....	364
<code>\TrackLangIfHasDefaultScript</code> .....	364
<code>\TrackLangRequireDialectPrefix</code> ..	364, 365

## U

<code>\u</code> .....	331
<code>\undef</code> .....	13, 14, 51, 186
<code>\underline</code> .....	188
<code>\unskip</code> .....	40, 51, 367
<code>upgreek</code> package .....	334
<code>\usepackage</code> .....	132, 133

## W

<code>\warn@nomakeglossaries</code> .....	117
<code>\warn@noprintglossary</code> .....	117, 120
<code>wrglossary</code> (counter) .....	23
<code>\write</code> .....	47, 103, 111, 116, 129, 130, 136

## X

<code>\x</code> .....	128, 152
<code>\xcapitalisewords</code> .....	173
<code>\xdef</code> .....	120, 141
<code>\xifinlist</code> .....	107
<code>\xifinlistcs</code> .....	32
<code>xindy</code> .....	408
<code>xindy</code> .....	14, 115
<code>xkeyval</code> package .....	5
<code>\XKV@checkchoice</code> .....	61
<code>\XKV@plfalse</code> .....	61
<code>\XKV@resa</code> .....	61
<code>\XKV@sttrue</code> .....	61