

`glossaries-extra.sty v1.27: documented code`

Nicola L.C. Talbot

Dickimaw Books

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

2018-02-26

Abstract

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

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

Contents

1	Main Package Code (glossaries-extra.sty)	5
1.1	Package Initialisation and Options	5
1.2	Extra Utilities	25
1.3	Modifications to Commands Provided by glossaries	33
1.3.1	Existence Checks	37
1.3.2	Document Definitions	45
1.3.3	Existing Glossary Style Modifications	50
1.3.4	Entry Formatting, Hyperlinks and Indexing	54
1.3.5	Entry Counting	89
1.3.6	Acronym Modifications	102
1.3.7	Indexing and Displaying Glossaries	105
1.4	Link Counting	140
1.5	Integration with glossaries-accsupp	142
1.6	Categories	153
1.7	Abbreviations	178
1.7.1	Abbreviation Styles Setup	197
1.7.2	Predefined Styles (Default Font)	200
1.7.3	Predefined Styles (Small Capitals)	217
1.7.4	Predefined Styles (Fake Small Capitals)	231
1.7.5	Predefined Styles (Emphasized)	245
1.7.6	Predefined Styles (User Parentheses Hook)	267
1.7.7	Predefined Styles (Hyphen)	276
1.7.8	Predefined Styles (No Short on First Use)	290
1.8	Using Entries in Headings	293
1.9	Multi-Lingual Support	312
1.10	glossaries-extra-bib2gls.sty	313
2	Style Adjustments (glossaries-extra-stylemods.sty)	347
2.1	Package Initialisation	347
2.2	List-Like Styles	348
2.3	Longtable Styles	351
2.4	Long Ragged Styles	353
2.5	Supertabular Styles	355
2.6	Super Ragged Styles	357
2.7	Inline Style	359
2.8	Tree Styles	359
2.9	Multicolumn Styles	376

3 bookindex style (glossary-bookindex.sty)	382
3.1 Package Initialisation and Options	382
Glossary	388
Change History	389
Index	406

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/02/26 v1.27 (NLCT)]

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

Requires etoolbox package.
4 \RequirePackage{etoolbox}

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

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

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

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

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

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

```

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.

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

`\glstr@recordsee` Does nothing by default.

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

`\ultnumberformat`

```
78 \newcommand*{\@glstr@defaultnumberformat}{glsnumberformat}%

```

`\ultNumberFormat`

```

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

```

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.

```

82 \newcommand*{\@glxtr@do@record@wrglossary}[1]{%
83 \begingroup
84 \ifKV@glslink@noindex
85 \else
86 \edef\@gls@label{\glsdetoklabel{#1}}%
87 \let\glslabel\@gls@label
88 \glswriteentry{#1}%
89 {%
90 \ifdefempty{\@glxtr@thevalue}%
91 {%
92 \ifx\@glxtr@org@theHvalue\@glxtr@theHvalue
93 \else
94 \let\theHglentrycounter\@glxtr@theHvalue
95 \fi
96 \glxtr@saveentrycounter
97 \let\@do@wrglossary\@glxtr@dorecord
98 }%
99 {%
100 \let\theHglentrycounter\@glxtr@thevalue
101 \let\theHglentrycounter\@glxtr@theHvalue
102 \let\@do@wrglossary\@glxtr@dorecordnodefer
103 }%
104 \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
105 \glxtr@do@wrglossary{#1}%
106 \else
107 \@glxtrwrglossmark
108 \@do@wrglossary
109 \fi
110 }%
111 \fi
112 \endgroup
113 }

```

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

```

114 \newcommand*{\glxtr@do@alsoindex@wrglossary}[1]{%
115 \glxtr@do@wrglossary{#1}%
116 \glxtr@do@record@wrglossary{#1}%
117 }

```

`@glxtr@record` The `record=only` option sets `\@glxtr@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`).

```

118 \newcommand*{\@glxtr@record}[3]{%
119 \ifglentryexists{#2}{}%
120 {%
121 \@glxtrwrglossmark

```


122 \beginngroup

Save the label in case it's needed.

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

Entry hasn't been defined, so we'll have to assume the page number by default.

```
129     \def\@gls@counter{page}%
```

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

```
130     \@gls@setdefault@glslink@opts
131     \setkeys{#3}{#1}%
132     \ifKV@glslink@noindex
133     \else
134     \glswriteentry{#2}%
135     {%
```

Check if thevalue has been set.

```
136     \ifdefempty{\@glsxtr@thevalue}%
137     {%
```

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

```
138     \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue
139     \else
140     \let\theHglentrycounter\@glsxtr@theHvalue
141     \fi
```

Save the entry counter.

```
142     \glsxtr@saveentrycounter
```

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

```
143     \let\@@do@@wrglossary\@glsxtr@dorecord
144     }%
145     {%
```

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.)

```
146     \let\theHglentrycounter\@glsxtr@thevalue
147     \let\theHglentrycounter\@glsxtr@theHvalue
148     \let\@@do@@wrglossary\@glsxtr@dorecordnodefer
149     }%
150     \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
151     \glsxtr@@do@@wrglossary{#2}%
152     \else
```

No need to escape special characters.

```
153     \@@do@@wrglossary
154     \fi
```

```

155     }%
156   \fi
157 \endgroup
158 }%
159 }

```

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

```

160 \newcommand*\@glsxtr@dorecord{%
161   \global\let\@glsrecordlocref\theglsentrycounter
162   \let\@glsxtr@orgprefix\@glo@counterprefix
163   \ifx\theglsentrycounter\theHglentrycounter
164     \def\@glo@counterprefix{}%
165   \else
166     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
167       {\theglsentrycounter}{\theHglentrycounter}}%
168     }%
169     \@do@gls@getcounterprefix
170   \fi
171   \protected@write\@auxout{\let\@glsrecordlocref\relax}{\string\glsxtr@record
172     {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}}%
173     {\@glsrecordlocref}}%
174   \@glsxtr@counterrecordhook
175   \let\@glo@counterprefix\@glsxtr@orgprefix
176 }

```

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

```

177 \newcommand*\@glsxtr@dorecordnodefer{%
178   \ifx\theglsentrycounter\theHglentrycounter
179     \protected@write\@auxout{{\string\glsxtr@record
180       {\@gls@label}{\@gls@counter}{\@glsnumberformat}}%
181       {\theglsentrycounter}}%
182   \else
183     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
184       {\theglsentrycounter}{\theHglentrycounter}}%
185     }%
186     \@do@gls@getcounterprefix
187     \protected@write\@auxout{{\string\glsxtr@record
188       {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}}%
189       {\theglsentrycounter}}%
190   \fi
191   \@glsxtr@counterrecordhook
192 }

```

`r@recordcounter`

```

193 \newcommand*\@@glsxtr@recordcounter{%
194   \@glsxtr@noop@recordcounter

```

```

195 }

p@recordcounter
196 \newcommand*{\@glxtr@noop@recordcounter}[1]{%
197   \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
198     requires record=only or record=alsoindex package option}{}%
199 }

p@recordcounter
200 \newcommand*{\@glxtr@op@recordcounter}[1]{%
201   \eappto\@glxtr@counterrecordhook{\noexpand\@glxtr@docounterrecord{#1}}%
202 }

lsxtr@recordsee Deal with \glssee in record mode.
203 \newcommand*{\@glxtr@recordsee}[2]{%
204   \@@glxtrwrglossmark
205   \def\@gls@xref{#2}%
206   \@onelevel@sanitize\@gls@xref
207   \protected@write\@auxout{}\{\string\glxtr@recordsee{#1}-\@gls@xref}\}%
208 }

srtglossaryunit
209 \newcommand{\printunsrtglossaryunit}{%
210   \print@noop@unsrtglossaryunit
211 }

tr@setup@record Initialise.
212 \newcommand*{\glxtr@setup@record}{\let\@do@wrglossary\glxtr@do@wrglossary}

aveentrycounter Only store the entry counter information if the indexing is on.
213 \newcommand*{\glxtr@indexonly@saveentrycounter}{%
214   \ifKV@glslink@noindex
215   \else
216     \glxtr@saveentrycounter
217   \fi
218 }

addloclistfield
219 \newcommand*{\glxtr@addloclistfield}{%
220   \key@ifundefined{glossentry}{loclist}%
221   {%
222     \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
223     \appto\@gls@keymap{,{loclist}{loclist}}%
224     \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
225     \appto\@newglossaryentryposthook{%
226       \gls@assign@field{\@glo@label}{loclist}{\@glo@loclist}%
227     }%
228     \glssetnoexpandfield{loclist}%
229   }%
230   {%

```

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

```

231 \key@ifundefined{glossentry}{location}%
232 {%
233   \define@key{glossentry}{location}{\def\@glo@location{##1}}%
234   \appto\@gls@keymap{,{location}{location}}%
235   \appto\@newglossaryentryprehook{\def\@glo@location{}}%
236   \appto\@newglossaryentryposthook{%
237     \gls@assign@field{\@glo@label}{location}{\@glo@location}%
238   }%
239   \glssetnoexpandfield{location}%
240 }%
241 {}%
```

Add a key to store the group heading.

```

242 \key@ifundefined{glossentry}{group}%
243 {%
244   \define@key{glossentry}{group}{\def\@glo@group{##1}}%
245   \appto\@gls@keymap{,{group}{group}}%
246   \appto\@newglossaryentryprehook{\def\@glo@group{}}%
247   \appto\@newglossaryentryposthook{%
248     \gls@assign@field{\@glo@label}{group}{\@glo@group}%
249   }%
250   \glssetnoexpandfield{group}%
251 }%
252 {}%
253 }
```

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

```

254 \newcommand*{\@glsxtr@record@setting}{off}
```

alsoindex

```

255 \newcommand*{\@glsxtr@record@setting@alsoindex}{alsoindex}
```

only

```

256 \newcommand*{\@glsxtr@record@setting@only}{only}
```

off

```

257 \newcommand*{\@glsxtr@record@setting@off}{off}
```

Now define the record package option.

```

258 \define@choicekey{glossaries-extra.sty}{record}[\val\nr]%
259 {off,only,alsoindex}%
260 [only]%
261 {%
262   \let\@glsxtr@record@setting\val
263   \ifcase\nr\relax
```

Don't record.

```

264   \def\glsxtr@setup@record{%
```

```

265 \renewcommand*{\@do@seeglossary}{\@glxtr@doseeglossary}%
266 \renewcommand*{\@glxtr@record}[3]{}%
267 \let\@do@wrglossary\glxtr@do@wrglossary
268 \let\@glxtr@saveentrycounter\glxtr@indexonly@saveentrycounter
269 \let\glxtrundefaction\glxtr@err@undefaction
270 \let\glxtr@warnonexistsordo\@gobble
271 \let\@glxtr@recordcounter\glxtr@noop@recordcounter
272 \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
273 \undef\glxtrsetaliasnoindex
274 }%
275 \or

```

Only record (don't index).

```

276 \def\glxtr@setup@record{%
277 \glxtr@autoseeindexfalse
278 \let\@do@seeglossary\glxtr@recordsee
279 \let\@glxtr@record\@glxtr@record
280 \let\@do@wrglossary\glxtr@do@record@wrglossary
281 \let\@glxtr@saveentrycounter\relax
282 \let\glxtrundefaction\glxtr@warn@undefaction
283 \let\glxtr@warnonexistsordo\glxtr@warn@onexistsordo
284 \glxtr@addloclistfield
285 \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
286 \let\@glxtr@recordcounter\glxtr@op@recordcounter
287 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%

```

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

```

288 \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.

```

289 \ifdefined\@glxtr@setupsort@none{\@glxtr@setupsort@none}{}%

```

Load glossaries-extra-bib2gls:

```

290 \RequirePackage{glossaries-extra-bib2gls}%
291 }%
292 \or

```

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

```

293 \def\glxtr@setup@record{%
294 \renewcommand*{\@do@seeglossary}{\@glxtr@dosee@alsoindex@glossary}%
295 \let\@glxtr@record\@glxtr@record
296 \let\@do@wrglossary\glxtr@do@alsoindex@wrglossary
297 \let\@glxtr@saveentrycounter\glxtr@indexonly@saveentrycounter
298 \let\glxtrundefaction\glxtr@warn@undefaction
299 \let\glxtr@warnonexistsordo\glxtr@warn@onexistsordo
300 \glxtr@addloclistfield
301 \let\@glxtr@recordcounter\glxtr@op@recordcounter
302 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%

```

```

303      \undef\glxtrsetaliasnoindex
304    }%
305  \fi
306 }

```

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.

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

```

307 \newcount\@glxtr@docdefval

```

Need to provide conditional commands that are backward compatible:

```

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

lsxtrdocdeftrue
309 \newcommand*\@glxtrdocdeftrue{\@glxtr@docdefval=1 }

sxtrdocdeffalse
310 \newcommand*\@glxtrdocdeffalse{\@glxtr@docdefval=0 }

```

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

```

311 \define@choicekey{glossaries-extra.sty}{docdef}[\val\nr]%
312 {false,true,restricted}[true]%
313 {%
314   \@glxtr@docdefval=\nr\relax
315   \ifnum\@glxtr@docdefval=2\relax
316     \renewcommand*\@glsdofexistsorwarn{\glsdofexists}%
317   \fi
318 }

```

`\docdefrestricted`

```

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

```

`\glsdofexistsorwarn` 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).

```

320 \newcommand*\@glsdofexistsorwarn{\glsdofexistsorwarn}

```

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

```

321 \define@boolkey{glossaries-extra.sty}{@glxtr}{indexcrossrefs}[true]{%
322   \if@glxtrindexcrossrefs
323   \else
324     \renewcommand*\@glxtr@autoindexcrossrefs{}}%
325 \fi
326 }

```

Switch off since this can increase the build time.

```
327 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

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

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

```
329 \define@boolkey{glossaries-extra.sty}{@glxtr@}{autoseeindex}[true]{%
330 }
331 \@glxtr@autoseeindextrue
```

iesExtraWarning Allow users to suppress warnings.

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

raWarningNoLine Allow users to suppress warnings.

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

335 \@glxtr@declareoption{nowarn}{%
336 \let\GlossariesExtraWarning\@gobble
337 \let\GlossariesExtraWarningNoLine\@gobble
338 \glxtr@doption{nowarn}%
339 }
```

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

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

postdot Shortcut for nopostdot=false

```
341 \@glxtr@declareoption{postdot}{%
342 \glxtr@doption{nopostdot=false}%
343 \renewcommand*{\@glxtr@defpostpunc}{%
344 \renewcommand*{\glspostdescription}{%
345 \ifglsnopostdot\else.\spacefactor\sfcode‘\.\ \fi}%
346 }%
347 }
```

nopostdot Needs to redefine \@glxtr@defpostpunc

```
348 \define@choicekey{glossaries-extra.sty}{nopostdot}{true,false}[true]{%
349 \glxtr@doption{nopostdot=#1}%
350 \renewcommand*{\@glxtr@defpostpunc}{%
351 \renewcommand*{\glspostdescription}{%
352 \ifglsnopostdot\else.\spacefactor\sfcode‘\.\ \fi}%
353 }%
354 }
```

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

```

355 \define@key{glossaries-extra.sty}{postpunc}{%
356   \glstr@dooption{nopostdot=false}%
357   \ifstrequal{#1}{dot}%
358   {%
359     \renewcommand*{\@glstr@defpostpunc}{%
360       \renewcommand*{\glspostdescription}{.\spacefactor\sfcode'\. }%
361     }%
362   }%
363   {%
364     \ifstrequal{#1}{comma}%
365     {%
366       \renewcommand*{\@glstr@defpostpunc}{%
367         \renewcommand*{\glspostdescription}{,}%
368       }%
369     }%
370     {%
371       \ifstrequal{#1}{none}%
372       {%
373         \glstr@dooption{nopostdot=true}%
374         \renewcommand*{\@glstr@defpostpunc}{%
375           \renewcommand*{\glspostdescription}{}%
376         }%
377       }%
378       {%
379         \renewcommand*{\@glstr@defpostpunc}{%
380           \renewcommand*{\glspostdescription}{#1}%
381         }%
382       }%
383     }%
384   }%
385 }
```

glstrabbrvtype Glossary type for abbreviations.

```

386 \newcommand*{\glstrabbrvtype}{\gldefaulttype}
```

abbreviationsdef Set by abbreviations option.

```

387 \newcommand*{\@glstr@abbreviationsdef}{}%
```

abbreviationsdef

```

388 \newcommand*{\@glstr@doabbreviationsdef}{%
389   \ifpackageloaded{babel}%
390   {\providecommand{\abbreviationsname}{\acronymname}}%
391   {\providecommand{\abbreviationsname}{Abbreviations}}%
392   \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
393   \renewcommand*{\glstrabbrvtype}{abbreviations}%
394   \newcommand*{\printabbreviations}[1][1]{%
395     \printglossary[type=\glstrabbrvtype,##1]%
396   }
```



```

396 }%
397 \disable@keys{glossaries-extra.sty}{abbreviations}%
    If the acronym option hasn't been used, change \acronymtype to \glxstrabbrvtype.
398 \ifglxacronym
399 \else
400   \renewcommand*{\acronymtype}{\glxstrabbrvtype}%
401 \fi
402 }%

```

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

```

403 \@glxstr@declareoption{abbreviations}{%
404   \let\@glxstr@abbreviationsdef\@glxstr@doabbreviationsdef
405 }

```

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).

```

406 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
407   \newcommand*{\ab}{\cgl{s}}%
408   \newcommand*{\abp}{\cgl{s}pl}%
409   \newcommand*{\as}{\glxstrshort}%
410   \newcommand*{\asp}{\glxstrshortpl}%
411   \newcommand*{\al}{\glxstrlong}%
412   \newcommand*{\alp}{\glxstrlongpl}%
413   \newcommand*{\af}{\glxstrfull}%
414   \newcommand*{\afp}{\glxstrfullpl}%
415   \newcommand*{\Ab}{\cGls}%
416   \newcommand*{\Abp}{\cGlspl}%
417   \newcommand*{\As}{\Glsxtrshort}%
418   \newcommand*{\Asp}{\Glsxtrshortpl}%
419   \newcommand*{\Al}{\Glsxtrlong}%
420   \newcommand*{\Alp}{\Glsxtrlongpl}%
421   \newcommand*{\Af}{\Glsxtrfull}%
422   \newcommand*{\Afp}{\Glsxtrfullpl}%
423   \newcommand*{\AB}{\cGLS}%
424   \newcommand*{\ABP}{\cGLSpl}%
425   \newcommand*{\AS}{\GLSxtrshort}%
426   \newcommand*{\ASP}{\GLSxtrshortpl}%
427   \newcommand*{\AL}{\GLSxtrlong}%
428   \newcommand*{\ALP}{\GLSxtrlongpl}%
429   \newcommand*{\AF}{\GLSxtrfull}%
430   \newcommand*{\AFP}{\GLSxtrfullpl}%
431   \providecommand*{\newabbr}{\newabbreviation}%

```

Disable this command after it's been used.

```

432 \let\GlsXtrDefineAbbreviationShortcuts\relax
433 }

```

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

```

434 \newcommand*{\GlsXtrDefineAcShortcuts}{%
435   \newcommand*{\ac}{\cglS}%
436   \newcommand*{\acp}{\cglSpl}%
437   \newcommand*{\acs}{\glSxtrshort}%
438   \newcommand*{\acsp}{\glSxtrshortpl}%
439   \newcommand*{\acl}{\glSxtrlong}%
440   \newcommand*{\aclp}{\glSxtrlongpl}%
441   \newcommand*{\acf}{\glSxtrfull}%
442   \newcommand*{\acfp}{\glSxtrfullpl}%
443   \newcommand*{\Ac}{\cGlS}%
444   \newcommand*{\Acp}{\cGlSpl}%
445   \newcommand*{\Acs}{\GlSxtrshort}%
446   \newcommand*{\Acsp}{\GlSxtrshortpl}%
447   \newcommand*{\Acl}{\GlSxtrlong}%
448   \newcommand*{\Aclp}{\GlSxtrlongpl}%
449   \newcommand*{\Acf}{\GlSxtrfull}%
450   \newcommand*{\Acfp}{\GlSxtrfullpl}%
451   \newcommand*{\AC}{\cGLS}%
452   \newcommand*{\ACP}{\cGLSpl}%
453   \newcommand*{\ACS}{\GLSxtrshort}%
454   \newcommand*{\ACSP}{\GLSxtrshortpl}%
455   \newcommand*{\ACL}{\GLSxtrlong}%
456   \newcommand*{\ACLP}{\GLSxtrlongpl}%
457   \newcommand*{\ACF}{\GLSxtrfull}%
458   \newcommand*{\ACFP}{\GLSxtrfullpl}%

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

```

Disable this command after it's been used.

```

460   \let\GlsXtrDefineAcShortcuts\relax
461 }

```

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

```

462 \newcommand*{\GlsXtrDefineOtherShortcuts}{%
463   \newcommand*{\newentry}{\newglossaryentry}%
464   \ifdef\printsymbols
465   {%
466     \newcommand*{\newsym}{\glSxtrnewsymbol}%
467   }{}%
468   \ifdef\printnumbers
469   {%
470     \newcommand*{\newnum}{\glSxtrnewnumber}%
471   }{}%
472   \let\GlsXtrDefineOtherShortcuts\relax
473 }

```

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.

```
474 \newcommand*{\@glxtr@setupshortcuts}{}
```

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

```
475 \newcommand*{\@glxtr@shortcutsval}{\ifglacrshortcuts acro\else none\fi}%
```

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

```
476 \define@choicekey{glossaries-extra.sty}{shortcuts}[\val\nr]%
477 {acronyms,acro,abbreviations,abbr,other,all,true,ac,none,false}[true]{%
478   \let\@glxtr@shortcutsval\val
479   \ifcase\nr\relax % acronyms
480     \renewcommand*{\@glxtr@setupshortcuts}{%
481       \glacrshortcutstrue
482       \DefineAcronymSynonyms
483     }%
484   \or % acro
485     \renewcommand*{\@glxtr@setupshortcuts}{%
486       \glacrshortcutstrue
487       \DefineAcronymSynonyms
488     }%
489   \or % abbreviations
490     \renewcommand*{\@glxtr@setupshortcuts}{%
491       \GlsXtrDefineAbbreviationShortcuts
492     }%
493   \or % abbr
494     \renewcommand*{\@glxtr@setupshortcuts}{%
495       \GlsXtrDefineAbbreviationShortcuts
496     }%
497   \or % other
498     \renewcommand*{\@glxtr@setupshortcuts}{%
499       \GlsXtrDefineOtherShortcuts
500     }%
501   \or % all
502     \renewcommand*{\@glxtr@setupshortcuts}{%
503       \glacrshortcutstrue
504       \GlsXtrDefineAcShortcuts
505       \GlsXtrDefineAbbreviationShortcuts
506       \GlsXtrDefineOtherShortcuts
507     }%
508   \or % true
509     \renewcommand*{\@glxtr@setupshortcuts}{%
```

```

510      \glsacrshortcutstrue

511      \GlsXtrDefineAcShortcuts
512      \GlsXtrDefineAbbreviationShortcuts
513      \GlsXtrDefineOtherShortcuts
514  }%

515  \or % ac
516      \renewcommand*{\@glxtr@setupshortcuts}{%
517      \glsacrshortcutstrue
518      \GlsXtrDefineAcShortcuts
519  }%

  Leave none and false as last option.

520  \else % none, false
521      \renewcommand*{\@glxtr@setupshortcuts}{}%
522  \fi
523  }

\lsxtr@doaccsupp
524 \newcommand*{\@glxtr@doaccsupp}{}

accsupp  If accsupp, load glossaries-accsupp package.
525 \@glxtr@declareoption{accsupp}{%
526 \renewcommand*{\@glxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

GlossaryWarning  Warning text displayed in document if the external glossary file given by the argument is miss-
ing.
527 \newcommand{\glxtrNoGlossaryWarning}[1]{%
528 \@glxtr@defaultnoglossarywarning{#1}%
529 }

nomissingglstext  If true, suppress the text produced if the external glossary file is missing.
530 \define@choicekey{glossaries-extra.sty}{nomissingglstext}[\val\nr]%
531 {true,false}[true]{%
532 \ifcase\nr\relax % true
533 \renewcommand{\glxtrNoGlossaryWarning}[1]{%
534 \null
535 }%
536 \else % false
537 \renewcommand{\glxtrNoGlossaryWarning}[1]{%
538 \@glxtr@defaultnoglossarywarning{#1}%
539 }%
540 \fi
541 }

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

xtr@redefstyles
542 \newcommand*{\@glxtr@redefstyles}{}

```

stylemods

```

543 \define@key{glossaries-extra.sty}{stylemods}[default]{%
544   \ifstrequal{#1}{default}%
545   {%
546     \renewcommand*{\@glxtr@redefstyles}{%
547       \RequirePackage{glossaries-extra-stylemods}}%
548   }%
549   {%
550     \ifstrequal{#1}{all}%
551     {%
552       \renewcommand*{\@glxtr@redefstyles}{%
553         \PassOptionsToPackage{all}{glossaries-extra-stylemods}%
554         \RequirePackage{glossaries-extra-stylemods}%
555       }%
556     }%
557     {%
558       \renewcommand*{\@glxtr@redefstyles}{}%
559       \@for\@glxtr@tmp:=#1\do{%
560         \IfFileExists{glossary-\@glxtr@tmp.sty}%
561         {%
562           \eappto\@glxtr@redefstyles{%
563             \noexpand\RequirePackage{glossary-\@glxtr@tmp}}%
564         }%
565         {%
566           \PackageError{glossaries-extra}%
567             {Glossaries style package ‘glossary-\@glxtr@tmp.sty’
568              doesn’t exist (did you mean to use the ‘style’ key?)}%
569             {The list of values (1) in the ‘stylemods’ key should
570              match the glossary-xxx.sty files provided with
571              glossaries.sty}%
572         }%
573       }%
574       \appto\@glxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
575     }
576   }%
577 }

```

glxtr@do@style

```

578 \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.

```

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

```

Defer actual style change:

```

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

```

Set this as the default style:

```

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

```

Set this style:

```
582 \setglossarystyle{#1}%  
583 }%  
584 }
```

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

```
585 \newcommand*{\@glstrwrglossmark}{}
```

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

```
586 \newcommand*{\@glstrwrglossmark}{}  
587 \AtBeginDocument{\renewcommand*{\@glstrwrglossmark}{\@glstrwrglossmark}}
```

`\glstrwrglossmark` Does nothing by default.

```
588 \newcommand*{\glstrwrglossmark}{\ensuremath{\cdot}}
```

`debug` Provide extra debug options.

```
589 \define@choicekey{glossaries-extra.sty}{debug}[\val\nr]%  
590 {true,false,showtargets,showwrgloss,all}[true]{%  
591 \ifcase\nr\relax % true  
592 \glstr@doption{debug=true}%  
593 \renewcommand*{\@glstrwrglossmark}{}%  
594 \or % false  
595 \glstr@doption{debug=false}%  
596 \renewcommand*{\@glstrwrglossmark}{}%  
597 \or % showtargets  
598 \glstr@doption{debug=showtargets}%  
599 \or % showwrgloss  
600 \glstr@doption{debug=true}%  
601 \renewcommand*{\@glstrwrglossmark}{\glstrwrglossmark}%  
602 \or % all  
603 \glstr@doption{debug=showtargets}%  
604 \renewcommand*{\@glstrwrglossmark}{\glstrwrglossmark}%  
605 \fi  
606 }
```

Pass all other options to `glossaries`.

```
607 \DeclareOptionX*{%  
608 \expandafter\glstr@doption\expandafter{\CurrentOption}}
```

Process options.

```
609 \ProcessOptionsX
```

Load `glossaries` if not already loaded.

```
610 \RequirePackage{glossaries}
```

Load the `glossaries-accsupp` package if required.

```
611 \@glstr@doaccsupp
```

Redefine \glspostdescription if required.

```
612 \@glstr@defpostpunc
```

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

```
613 \def\glsshowtarget#1{%
614   \glstrtitleorpdforheading
615   {%
616     \ifmmode
617       \texttt{\small [#1]}%
618     \else
619       \ifinner
620         \texttt{\small [#1]}%
621       \else
622         \marginpar{\texttt{\small #1}}%
623       \fi
624     \fi
625   }%
626   {[#1]}%
627   {\texttt{\small [#1]}}%
628 }
```

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

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

r@doseeglossary

```
630 \newcommand*{\@glstr@doseeglossary}[2]{%
631   \glsdodefexists{#1}%
632   {%
633     \@glstrwrglossmark
634     \@glstr@org@doseeglossary{#1}{#2}%
635   }%
636 }
```

oindex@glossary

```
637 \newcommand*{\@glstr@dosee@alsoindex@glossary}[2]{%
638   \@glstr@recordsee{#1}{#2}%
639   \@glstr@doseeglossary{#1}{#2}%
640 }
```

@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.)

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

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

```
642 \if@glstr@autoseeindex
643 \else
```

```

644 \ifdef\@glxtr@org@gloautosee
645 {}%
646 {\PackageError{glossaries-extra}{‘autoseeindex=false’ package
647 option requires at least v4.30 of glossaries.sty}%
648 {You need to update the glossaries.sty package}%
649 }
650 \fi

```

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

```

651 \ifdef\@glo@autosee
652 {%
653 \renewcommand*{\@glo@autosee}{%
654 \if@glxtr@autoseeindex\@glxtr@org@gloautosee\fi}%
655 }%
656 {}

```

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.

```

657 \renewcommand*{\gls@checkseeallowed}{%
658 \if@glxtr@autoseeindex\@gls@see@noindex\fi
659 }

```

Define abbreviations glossaries if required.

```

660 \@glxtr@abbreviationsdef
661 \let\@glxtr@abbreviationsdef\relax

```

Setup shortcuts if required.

```

662 \@glxtr@setupshortcuts

```

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

```

663 \@glxtr@redef@for@gl@sentries

```

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

```

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

```

Now define the user command:

```

665 \newcommand*{\glossariesextrasetup}[1]{%
666 \let\glxtr@setup@record\relax
667 \let\@glxtr@setupshortcuts\relax
668 \let\@glxtr@redef@for@gl@sentries\relax
669 \setkeys{glossaries-extra.sty}{#1}%
670 \@glxtr@abbreviationsdef
671 \let\@glxtr@abbreviationsdef\relax
672 \@glxtr@setupshortcuts
673 \glxtr@setup@record
674 \@glxtr@redef@for@gl@sentries
675 }

```


@@do@wrglossary Save original definition of @@do@wrglossary.
676 \let\glxtr@org@@do@wrglossary\@@do@wrglossary

@@do@wrglossary The new version adds code that can show a marker for debugging.
677 \newcommand*{\glxtr@@do@wrglossary}[1]{%
678 \@@glxtrwrglossmark
679 \glxtr@org@@do@wrglossary{#1}%
680 }

saveentrycounter Save original definition of \@gls@saveentrycounter.
681 \let\glxtr@saveentrycounter\@gls@saveentrycounter

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

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

sxtrdialecthook
683 \newcommand*{\@glxtrdialecthook}{}

Set up record option if required.

684 \glxtr@setup@record

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

685 \AtBeginDocument{%
686 \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
687 \def\@glxtrundeftag{\glxtrundeftag}%
688 }

1.2 Extra Utilities

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

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

689 \newcommand{\glxtrifemptyglossary}[3]{%
690 \ifcsdef{glolist@#1}%
691 {%
692 \ifcsstring{glolist@#1}{,}{#2}{#3}%

```

693 }%
694 {%
695   \glsxtrundefaction{Glossary type ‘#1’ doesn’t exist}{}%
696   #2%
697 }%
698 }

xtrifkeydefined Tests if the key given in the first argument has been defined.
699 \newcommand*{\glsxtrifkeydefined}[3]{%
700   \key@ifundefined{glossentry}{#1}{#3}{#2}%
701 }

providestoragekey Like \glxsaddstoragekey but does nothing if the key has already been defined.
702 \newcommand*{\glsxtrprovidestoragekey}{%
703   \@ifstar\@sglsxtr@provide@storagekey\@glsxtr@provide@storagekey
704 }

vide@storagekey Unstarred version.
705 \newcommand*{\@glsxtr@provide@storagekey}[3]{%
706   \key@ifundefined{glossentry}{#1}%
707   {%
708     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
709     \appto\@gls@keymap{,{#1}{#1}}%
710     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
711     \appto\@newglossaryentryposthook{%
712       \letcs{\@glo@tmp}{@glo@#1}%
713       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
714     }%

    Allow the user to omit the user level command if they only intended fetching the value with
    \glsxtrusefield
715     \ifblank{#3}
716     {}%
717     {%
718       \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
719     }%
720   }%
721   {%

    Provide the no-link command if not already defined.
722     \ifblank{#3}
723     {}%
724     {%
725       \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
726     }%
727   }%
728 }

vide@storagekey Starred version.

```

```

729 \newcommand*{\s@glxtr@provide@storagekey}[1]{%
730   \key@ifundefined{glossentry}{#1}%
731   {%
732     \expandafter\newcommand\expandafter*\expandafter
733     {\csname gls@assign@#1@field\endcsname}[2]{%
734       \@gls@expand@field{##1}{#1}{##2}%
735     }%
736   }%
737   {%
738     \@glxtr@provide@addstoragekey{#1}%
739   }

```

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`

```

740 \newcommand{\GlsXtrFmtField}{useri}

```

`tDefaultOptions`

```

741 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

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

```

`\@glxtrfmt` Unstarred form.

```

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

```

`\s@glxtrfmt` Starred form.

```

744 \newcommand*{\s@glxtrfmt}[3][\@glxtrfmt{#1}{#2}{#3}{}
745 \new@ifnextchar[\s@glxtrfmt{#1}{#2}{#3}{}%
746 {\@glxtrfmt{#1}{#2}{#3}{}%
747 }

```

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

```

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

```

`\@glxtrfmt` Actual inner working.

```

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

```

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

```

750 \begingroup
751   \def\glslabel{#2}%
752   \glsdoifexistsordo{#2}%
753   {%

```

```

754 \ifglshasfield{\GlsXtrFmtField}{#2}%
755 {%
756 \let\do@gl@link@checkfirsthyper\relax
757 \expandafter\@gl@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
758 {\glxtrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
759 }%
760 {\glxtrfmtdisplay{@firstofone}{#3}{#4}}%
761 }%
762 {%

```

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

```

763 \begingroup
764 \@gl@setdefault@gl@link@opts
765 \setkeys{gl@link}{\GlsXtrFmtDefaultOptions,#1}%
766 \ifKV@gl@link@noindex\else\gl@sadd{#2}\fi
767 \endgroup
768 \glxtrfmtdisplay{@firstofone}{#3}{#4}%
769 }%
770 \endgroup
771 }

```

glxtrfmtdisplay The command used internally by \glxtrfmt 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).

```

772 \newcommand{\glxtrfmtdisplay}[3]{\csuse{#1}{#2}#3}

```

glxtrentryfmt No link or indexing.

```

773 \ifdef\texorpdfstring
774 {
775 \newcommand*{\glxtrentryfmt}[2]{%
776 \texorpdfstring{\@glxtrentryfmt{#1}{#2}}{#2}%
777 }
778 }
779 {
780 \newcommand*{\glxtrentryfmt}{\@glxtrentryfmt}
781 }

```

@glxtrentryfmt

```

782 \newrobustcmd*{\@glxtrentryfmt}[2]{%
783 \gl@sdoifexistsordo{#1}%
784 {%
785 \ifglshasfield{\GlsXtrFmtField}{#1}%
786 {%
787 \csuse{\glscurrentfieldvalue}{#2}%
788 }%
789 {#2}%
790 }%
791 {#2}%

```

792 }

`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.

```
793 \newcommand*{\glxtrfieldlistadd}[3]{%
794   \listcsadd{glo@\glsdetoklabel{#1}@#2}{#3}%
795 }
```

`trfieldlistgadd` Similarly but uses `\listcsgadd`.

```
796 \newcommand*{\glxtrfieldlistgadd}[3]{%
797   \listcsgadd{glo@\glsdetoklabel{#1}@#2}{#3}%
798 }
```

`trfieldlisteadd` Similarly but uses `\listcseadd`.

```
799 \newcommand*{\glxtrfieldlisteadd}[3]{%
800   \listcseadd{glo@\glsdetoklabel{#1}@#2}{#3}%
801 }
```

`trfieldlistxadd` Similarly but uses `\listcsxadd`.

```
802 \newcommand*{\glxtrfieldlistxadd}[3]{%
803   \listcsxadd{glo@\glsdetoklabel{#1}@#2}{#3}%
804 }
```

Now provide commands to iterate over these lists.

`fielddolistloop`

```
805 \newcommand*{\glxtrfielddolistloop}[2]{%
806   \dolistcsloop{glo@\glsdetoklabel{#1}@#2}%
807 }
```

`fieldforlistloop`

```
808 \newcommand*{\glxtrfieldforlistloop}[3]{%
809   \forlistcsloop{glo@\glsdetoklabel{#1}@#2}{#3}%
810 }
```

List element tests:

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

```
811 \newcommand*{\glxtrfieldifinlist}[5]{%
812   \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
813 }
```

`trfieldxifinlist` Expands item.

```
814 \newcommand*{\glxtrfieldxifinlist}[5]{%
815   \xifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
816 }
```

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

```
817 \newcommand*{\glxtrforcsvfield}[3]{%
818   \@glxtrifhasfield{#2}{#1}%
819   {%
820     \let\glxtrendfor\@endfortrue
821     \@for\@glxtr@label:=\glscurrentfieldvalue\do
822       {\expandafter#3\expandafter{\@glxtr@label}}}%
823   }%
824 }
```

`\glxtrifhasfield` 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.

```
825 \newrobustcmd{\glxtrifhasfield}{%
826   \@ifstar{\s@glxtrifhasfield}{\@glxtrifhasfield}%
827 }
```

`\glxtrifhasfield` Unstarred version adds grouping.

```
828 \newcommand{\@glxtrifhasfield}[4]{%
829   {\s@glxtrifhasfield{#1}{#2}{#3}{#4}}%
830 }
```

`\glxtrifhasfield` Starred version omits grouping.

```
831 \newcommand{\s@glxtrifhasfield}[4]{%
832   \letcs{\glscurrentfieldvalue}{glo@\glsdetoklabel{#2}@#1}%
833   \ifundef\glscurrentfieldvalue
834     {#4}%
835     {%
836       \ifdefempty\glscurrentfieldvalue{#4}{#3}%
837     }%
838 }
```

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

Just uses `\ifcsundef`.

```
839 \newcommand{\GlsXtrIfFieldUndef}[2]{%
840   \ifcsundef{glo@\glsdetoklabel{#2}@#1}%
841 }
```

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

```
842 \newcommand*{\glxtrusefield}[2]{%
843   \@gls@entry@field{#1}{#2}%
844 }
```

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

```

845 \newcommand*{\Glsxtrusefield}[2]{%
846   \@Gls@entry@field{#1}{#2}%
847 }
```

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

```

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

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

```

849 \newcommand*{\glxstredeffield}[2]{\csedef{glo@\glsdetoklabel{#1}@#2}}
```

`etfieldifexists`

```

850 \newcommand*{\glxtrsetfieldifexists}[3]{\glsoifexists{#1}{#3}}
```

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

```

851 \newrobustcmd*{\GlsXtrSetField}[3]{%
852   \glxtrsetfieldifexists{#1}{#2}%
853   {\csdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
854 }
```

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

```

855 \newrobustcmd*{\GlsXtrLetField}[3]{%
856   \glxtrsetfieldifexists{#1}{#2}%
857   {\cslet{glo@\glsdetoklabel{#1}@#2}{#3}}%
858 }
```

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

```

859 \newrobustcmd*{\csGlsXtrLetField}[3]{%
860   \glxtrsetfieldifexists{#1}{#2}%
861   {\csletcs{glo@\glsdetoklabel{#1}@#2}{#3}}%
862 }
```

`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.

```

863 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
864   \glxtrsetfieldifexists{#1}{#2}%
865   {\csletcs{glo@\glsdetoklabel{#1}@#2}{glo@\glsdetoklabel{#3}@#4}}%
866 }
```

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

```

867 \newrobustcmd*{\gGlsXtrSetField}[3]{%
868   \glxtrsetfieldifexists{#1}{#2}%
869   {\csgdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
870 }
```

xGlsXtrSetField

```
871 \newrobustcmd*{\xGlsXtrSetField}[3]{%
872   \glstrsetfieldifexists{#1}{#2}%
873   {\protected@csxdef{glo\xGlsdetoklabel{#1}@#2}{#3}}%
874 }
```

eGlsXtrSetField

```
875 \newrobustcmd*{\eGlsXtrSetField}[3]{%
876   \glstrsetfieldifexists{#1}{#2}%
877   {\protected@csedef{glo\xGlsdetoklabel{#1}@#2}{#3}}%
878 }
```

XtrIfFieldEqStr

```
879 \newrobustcmd*{\GlsXtrIfFieldEqStr}[5]{%
880   \glstrifhasfield{#1}{#2}%
881   {%
882     \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
883   }%
884   {#5}%
885 }
```

\glstrpageref Like \glssrefentry but references the page number instead (if entry counting is on).

```
886 \ifglssentrycounter
887   \newcommand*{\glstrpageref}[1]{\pageref{glssentry-\Glsdetoklabel{#1}}}
888 \else
889   \ifglssubentrycounter
890     \newcommand*{\glstrpageref}[1]{\pageref{glssentry-\Glsdetoklabel{#1}}}
891   \else
892     \newcommand*{\glstrpageref}[1]{\Gls{#1}}
893   \fi
894 \fi
```

lossarypreamble

```
895 \newcommand{\apptoglossarypreamble}[2][\Glsdefaulttype]{%
896   \ifcsdef{glolist@#1}%
897   {%
898     \ifcsundef{@glossarypreamble@#1}%
899     {\csdef{@glossarypreamble@#1}{}}%
900   }%
901   \csappto{@glossarypreamble@#1}{#2}%
902   }%
903   {%
904     \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
905   }%
906 }
```

lossarypreamble

```
907 \newcommand{\preglossarypreamble}[2][\Glsdefaulttype]{%
```



```

908 \ifcsdef{glolist@#1}%
909 {%
910 \ifcsundef{@glossarypreamble@#1}%
911 {\csdef{@glossarypreamble@#1}{}}%
912 {}%
913 \cspretto{@glossarypreamble@#1}{#2}%
914 }%
915 {%
916 \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
917 }%
918 }

```

1.3 Modifications to Commands Provided by glossaries

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

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

`\newglossaryentry`

```

919 \renewcommand*{\longnewglossaryentry}{%
920 \@ifstar{\glstr@s@longnewglossaryentry\glstr@longnewglossaryentry
921 }

```

`\newglossaryentry` Starred version.

```

922 \newcommand{\@glstr@s@longnewglossaryentry}[3]{%
923 \glsdoifnoexists{#1}%
924 {%
925 \bgroup
926 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
927 \long\def\@newglossaryentryprehook{%
928 \long\def\@glo@desc{#3}%
929 \@org@newglossaryentryprehook
930 }%
931 \renewcommand*{\gls@assign@desc}[1]{%
932 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
933 \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
934 }
935 \gls@defglossaryentry{#1}{#2}%
936 \egroup
937 }%
938 }

```

`\newglossaryentry` Unstarred version.

```

939 \newcommand{\@glstr@longnewglossaryentry}[3]{%
940 \glsdoifnoexists{#1}%

```

```

941  {%
942    \bgroup
943      \let\@org@newglossaryentryprehook\@newglossaryentryprehook
944      \long\def\@newglossaryentryprehook{%
945        \long\def\@glo@desc{#3\glstrpostlongdescription}%
946        \@org@newglossaryentryprehook
947      }%
948      \renewcommand*{\gls@assign@desc}[1]{%
949        \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%

```

The following is different from the base glossaries.sty:

```

950      \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
951    }
952    \gls@defglossaryentry{#1}{#2}%
953  \egroup
954 }%
955 }

```

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

```

956 \newcommand*{\glstrpostlongdescription}{\leavevmode\unskip\nopostdesc}

```

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

`\ignoredglossary` Redefine to check for star.

```

957 \renewcommand{\newignoredglossary}{%
958   \@ifstar\glstr@s@newignoredglossary\glstr@org@newignoredglossary
959 }

```

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

```

960 \newcommand*{\glstr@org@newignoredglossary}[1]{%
961   \ifcsdef{glolist@#1}
962   {%
963     \glstrundefaction{Glossary type ‘#1’ already exists}{}%
964   }%
965   {%
966     \ifdefempty\@ignored@glossaries
967     {%
968       \edef\@ignored@glossaries{#1}%
969     }%
970     {%
971       \eappto\@ignored@glossaries{, #1}%
972     }%
973     \csgdef{glolist@#1}{,}%
974     \ifcsundef{gls@#1@entryfmt}%
975     {%
976       \defglsentryfmt[#1]{\glsentryfmt}%
977     }%
978   }%

```

```

979 \ifdefempty\@gls@nohyperlist
980 {%
981   \renewcommand*{\@gls@nohyperlist}{#1}%
982   }%
983   {%
984     \eappto\@gls@nohyperlist{, #1}%
985     }%
986   }%
987 }

```

ignoredglossary Starred form.

```

988 \newcommand*{\glstr@s@newignoredglossary}[1]{%
989   \ifcsdef{glolist@#1}
990   {%
991     \glstrundefaction{Glossary type ‘#1’ already exists}{}%
992   }%
993   {%
994     \ifdefempty\@ignored@glossaries
995     {%
996       \edef\@ignored@glossaries{#1}%
997     }%
998     {%
999       \eappto\@ignored@glossaries{, #1}%
1000     }%
1001     \csgdef{glolist@#1}{,}%
1002     \ifcsundef{gls@#1@entryfmt}%
1003     {%
1004       \defglsentryfmt[#1]{\glsentryfmt}%
1005     }%
1006     {}%
1007   }%
1008 }

```

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

```

1009 \glsifusetranslator
1010 {%
1011   \renewcommand*{\glissettoctitle}[1]{%
1012     \ifcsdef{gls@tr@set@#1@toctitle}%
1013     {%
1014       \csuse{gls@tr@set@#1@toctitle}%
1015     }%
1016     {%
1017       \ifcsdef{@glotype@#1@title}%
1018       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1019       {\def\glossarytoctitle{\glossarytitle}}%
1020     }%
1021   }%
1022 }

```

```

1023 {
1024   \renewcommand*{\glsettoctitle}[1]{%
1025     \ifcsdef{@glotype@#1@title}%
1026       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1027       {\def\glossarytoctitle{\glossarytitle}}%
1028   }
1029 }

```

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

```

1030 \newcommand{\provideignoredglossary}{%
1031   \@ifstar\glxtr@s@provideignoredglossary\glxtr@provideignoredglossary
1032 }

```

ignoredglossary Unstarred version.

```

1033 \newcommand*{\glxtr@provideignoredglossary}[1]{%
1034   \ifcsdef{glolist@#1}
1035   {}%
1036   {%
1037     \ifdefempty\@ignored@glossaries
1038     {%
1039       \edef\@ignored@glossaries{#1}%
1040     }%
1041     {%
1042       \eappto\@ignored@glossaries{,#1}%
1043     }%
1044     \csgdef{glolist@#1}{,}%
1045     \ifcsundef{gls@#1@entryfmt}%
1046     {%
1047       \def\glsentryfmt[#1]{\glsentryfmt}%
1048     }%
1049     {}%
1050     \ifdefempty\@gls@nohyperlist
1051     {%
1052       \renewcommand*{\@gls@nohyperlist}{#1}%
1053     }%
1054     {%
1055       \eappto\@gls@nohyperlist{,#1}%
1056     }%
1057   }%
1058 }

```

ignoredglossary Starred form.

```

1059 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
1060   \ifcsdef{glolist@#1}
1061   {}%
1062   {%
1063     \ifdefempty\@ignored@glossaries
1064     {%
1065       \edef\@ignored@glossaries{#1}%

```

```

1066 }%
1067 {%
1068   \eappto\@ignored@glossaries{,#1}%
1069 }%
1070 \csgdef{glolist@#1}{,}%
1071 \ifcsundef{gls@#1@entryfmt}%
1072 {%
1073   \defglsentryfmt[#1]{\glsentryfmt}%
1074 }%
1075 {}%
1076 }%
1077 }

```

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

```

1078 \newcommand*{\glstrcopytoglossary}[2]{%
1079   \glsdoidexists{#1}%
1080   {%
1081     \ifcsdef{glolist@#2}
1082     {%
1083       \cseappto{glolist@#2}{#1,}%
1084     }%
1085     {%
1086       \glstrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
1087     }%
1088   }%
1089 }

```

1.3.1 Existence Checks

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

```

1090 \renewcommand{\glsdoidexists}[2]{%
1091   \ifglsentryexists{#1}{#2}%
1092   {%

```

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

```

1093   \edef\glslabel{\glsdetoklabel{#1}}%
1094   \glstrundefaction{Glossary entry ‘\glslabel’
1095     has not been defined}{You need to define a glossary entry before
1096     you can reference it.}%
1097   }%
1098 }

```

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

```

1099 \renewcommand{\glsdoidnoexists}[2]{%
1100   \ifglsentryexists{#1}{%
1101     \glstrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
1102       has already been defined}{}}{#2}%

```

1103 }

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

```
1104 \ifdef\glsdoifexistsordo
1105 {%
1106   \renewcommand{\glsdoifexistsordo}[3]{%
1107     \ifglstryexists{#1}{#2}%
1108     {%
1109       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1110         has not been defined}{You need to define a glossary entry
1111         before you can use it.}%
1112       #3%
1113     }%
1114   }%
1115 }
1116 {%
1117   \glstr@warnonexistsordo\glsdoifexistsordo
1118   \newcommand{\glsdoifexistsordo}[3]{%
1119     \ifglstryexists{#1}{#2}%
1120     {%
1121       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
1122         has not been defined}{You need to define a glossary entry
1123         before you can use it.}%
1124       #3%
1125     }%
1126   }%
1127 }
```

`\glsarynoexistsordo` Similarly for `\doifglossarynoexistsordo`.

```
1128 \ifdef\doifglossarynoexistsordo
1129 {%
1130   \renewcommand{\doifglossarynoexistsordo}[3]{%
1131     \ifglossaryexists{#1}%
1132     {%
1133       \glstrundefaction{Glossary type '#1' already exists}{}%
1134       #3%
1135     }%
1136     {#2}%
1137   }%
1138 }
1139 {%
1140   \glstr@warnonexistsordo\doifglossarynoexistsordo
1141   \newcommand{\doifglossarynoexistsordo}[3]{%
1142     \ifglossaryexists{#1}%
1143     {%
1144       \glstrundefaction{Glossary type '#1' already exists}{}%
1145       #3%
1146     }%
1147   }%
1148 }
```

```

1147     {#2}%
1148   }%
1149 }
1150

```

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.

```

1151 \appto\@newglossaryentryposthook{%
1152   \ifdefvoid\@glo@see
1153     {\csxdef{glo@\@glo@label @see}{}}%
1154     {%
1155       \csxdef{glo@\@glo@label @see}{\@glo@see}%
1156       \ifglxtr@autoseeindex
1157         \@glxtr@autoindexcrossrefs
1158       \fi
1159     }%
1160 }
1161 \appto\@gls@keymap{,{see}{see}}

```

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

```

1162 \newcommand*{\glxtrusesee}[1]{%
1163   \glsdoifexists{#1}%
1164   {%
1165     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
1166     \ifdefempty\@glo@see
1167       {}%
1168     {%
1169       \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
1170     }%
1171   }%
1172 }

```

\glxtr@usesee

```

1173 \newcommand*{\glxtr@usesee}[1][\seename]{%
1174   \@glxtr@usesee{#1}%
1175 }

```

\@glxtr@usesee

```

1176 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
1177   \glxtruseseeformat{#1}{#2}%
1178 }

```

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.

```

1179 \newcommand*{\glxtruseseeformat}[2]{%

```

```

1180 \glsseeformat[#1]{#2}{}%
1181 }

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

1182 \renewcommand*{\glsseeitemformat}[1]{%
1183 \ifglshashshort{\glslabel}{\glsaccesstext{#1}}{\glsaccessname{#1}}}%
1184 }

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

1185 \newcommand*{\glsxtruseealso}[1]{%
1186 \glsdoifexists{#1}%
1187 {%
1188 \letcs{\@glo@see}{glo\glsdetoklabel{#1}@seealso}%
1189 \ifdefempty\@glo@see
1190 {}%
1191 {%
1192 \expandafter\glsxtruseealsoformat\expandafter{\@glo@see}%
1193 }%
1194 }%
1195 }

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

1196 \newcommand*{\glsxtruseealsoformat}[1]{%
1197 \glsseeformat[\seealso]{#1}{}%
1198 }

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

1199 \newrobustcmd{\glsxtrseelist}[1]{%
1200 \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
1201 }

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

1202 \providecommand{\seealso}{}

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

1203 \ifdef\@xdycrossrefhook
1204 {

```


Add the cross-reference class definition to the hook.

```

1205 \appto\@xdycrossrefhook{%
1206   \write\glswrite{(define-crossref-class \string"seealso\string"
1207     :unverified )}%
1208   \write\glswrite{(markup-crossref-list
1209     :class \string"seealso\string"^^J\space\space\space
1210     :open \string"\string\glxtruseealsoformat\glsopenbrace\string"
1211     :close \string"\glsclosebrace\string")}%
1212 }

```

Append to class list.

```

1213 \appto\@xdylocationclassorder{\space\string"seealso\string"}

```

This essentially works like `\@do@seeglossary` but uses the `seealso` class.

```

1214 \newrobustcmd*{\glxtrindexseealso}[2]{%
1215   \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
1216     \@glxtr@recordsee{#1}{#2}%
1217   \fi
1218   \glsdoifexists{#1}%
1219   {%
1220     \@@glxtrwrglossmark
1221     \def\@gls@xref{#2}%
1222     \@onelevel@sanitize\@gls@xref
1223     \@gls@checkmkidxchars\@gls@xref
1224     \gls@glossary{\csname glo@#1@type\endcsname}{%
1225       (indexentry
1226         :tkey (\csname glo@#1@index\endcsname)
1227         :xref (\string"\@gls@xref\string")
1228         :attr \string"seealso\string"
1229       )
1230     }%
1231   }%
1232 }
1233 }
1234 {

```

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

```

1235 \newrobustcmd*{\glxtrindexseealso}{\glssee[\seealsoname]}
1236 }

```

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`.

```

1237 \ifdef\gls@set@xr@key
1238 {

```

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

```

1239 \define@key{glossentry}{alias}{%
1240   \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1241 }
1242 \define@key{glossentry}{seealso}{%
1243   \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1244 }

```

Add to the key mappings.

```

1245 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}

```

Set the default value.

```

1246 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%

```

Assign the field values.

```

1247 \appto\@newglossaryentryposthook{%
1248   \ifdefvoid\@glo@seealso
1249     {\csxdef{glo@\@glo@label @seealso}{}}%
1250     {%
1251       \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
1252       \if@glxtr@autoseeindex
1253         \@glxtr@autoindexcrossrefs
1254       \fi
1255     }%

```

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

```

1256   \ifdefvoid\@glo@alias
1257     {\csxdef{glo@\@glo@label @alias}{}}%
1258     {%
1259       \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1260     }%
1261 }

```

Provide user-level commands to access the values.

`\glxtralias`

```

1262 \newcommand*{\glxtralias}[1]{\@gls@entry@field{#1}{alias}}

```

`trseealsolabels`

```

1263 \newcommand*{\glxtrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}

```

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

```

1264 \appto\@glo@autoseehook{%
1265   \ifdefvoid\@glo@alias
1266     {%
1267       \ifdefvoid\@glo@seealso
1268         {}%
1269       {%
1270         \edef\@do@glsee{\noexpand\glxtrindexseealso
1271           {\@glo@label}{\@glo@seealso}}%
1272         \@do@glsee

```

```

1273     }%
1274 }%
1275 {%

```

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

```

1276     \ifdefvoid\@glo@see
1277     {%
1278         \edef\@do@glsssee{\noexpand\glsssee{\@glo@label}{\@glo@alias}}%
1279         \@do@glsssee
1280     }%
1281     {}%
1282 }%
1283 }%
1284 }
1285 {

```

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

\glsxtralias

```

1286 \glsaddstoragekey*{alias}{\glsxtralias}

```

trseealsolabels

```

1287 \glsaddstoragekey*{seealso}{\glsxtrseealsolabels}

```

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

ryentryposthook Append to the hook to check for the alias and seealso keys.

```

1288 \appto\@newglossaryentryposthook{%
1289     \ifcsvoid{glo@\@glo@label @alias}%
1290     {%
1291         \ifcsvoid{glo@\@glo@label @seealso}%
1292         {}%
1293         {%
1294             \edef\@do@glsssee{\noexpand\glsxtrindexseealso
1295                 {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1296             \@do@glsssee
1297         }%
1298     }%
1299     {%

```

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

```

1300     \ifdefvoid\@glo@see
1301     {%
1302         \edef\@do@glsssee{\noexpand\glsssee
1303             {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1304         \@do@glsssee
1305     }%
1306     {}%
1307 }%
1308 }

```

1309 }

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

1310 \AtEndDocument{\if@glxtrindexcrossrefs\glxtraddallcrossrefs\fi}

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

```
1311 \newcommand*{\glxtraddallcrossrefs}{%
1312   \forallglossaries{\@glo@type}%
1313   {%
1314     \for@gl@entries[\@glo@type]{\@glo@label}%
1315     {%
1316       \if@gl@sused{\@glo@label}%
1317       {\expandafter\glxtr@addunusedxrefs\expandafter{\@glo@label}}}%
1318   }%
1319 }%
1320 }
```

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

```
1321 \newcommand*{\@glxtr@addunusedxrefs}[1]{%
1322   \letcs{\@glo@see}{glo@gl@detoklabel{#1}@see}%
1323   \ifdefvoid\@glo@see
1324   {%
1325     {%
1326       \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1327     }%
1328     \letcs{\@glo@see}{glo@gl@detoklabel{#1}@seealso}%
1329     \ifdefvoid\@glo@see
1330     {%
1331       {%
1332         \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1333       }%
1334     }
```

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

```
1335 \newcommand*{\glxtr@addunused}[1][]{%
1336   \@glxtr@addunused
1337 }
```

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

```
1338 \def\@glxtr@addunused#1\@end@glxtr@addunused{%
1339   \@for\@glxtr@label:=#1\do
1340   {%
1341     \if@gl@sused{\@glxtr@label}}}%
1342   {%
1343     \gl@sadd[format=glxtrunusedformat]{\@glxtr@label}%
1344     \gl@unset{\@glxtr@label}%
1345     \expandafter\glxtr@addunusedxrefs\expandafter{\@glxtr@label}%
1346   }
```

```

1346 }%
1347 }%
1348 }

```

xtrunusedformat

```

1349 \newcommand*{\glxtrunusedformat}[1]{\unskip}

```

1.3.2 Document Definitions

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

```

1350 \let\glxtr@orgmakenoidxglossaries\makenoidxglossaries
1351 \renewcommand{\makenoidxglossaries}{%
1352   \ifdefequal\@glxtr@record@setting\@glxtr@record@setting@off
1353   {%
1354     \glxtr@orgmakenoidxglossaries

```

Add marker to \do@seeglossary

```

1355   \renewcommand{\do@seeglossary}[2]{%
1356     \@glxtrwrglossmark
1357     \edef\@gls@label{\glsdetoklabel{##1}}%
1358     \protected@write\@auxout{}{%
1359       \string\@gls@reference
1360       {\csname glo@\@gls@label @type\endcsname}%
1361       {\@gls@label}%
1362       {%
1363         \string\glsseeformat##2}%
1364       }%
1365     }%
1366   }%

```

Check for docdefs=restricted:

```

1367   \if@glxtrdocdefrestricted

```

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

```

1368     \renewcommand*{\@gls@reference}[3]{%
1369       \ifcsundef{@glsref@##1}{\csgdef{@glsref@##1}{}}{}%
1370       \ifinlistcs{##2}{@glsref@##1}%
1371       {}%
1372       {\listcsgadd{@glsref@##1}{##2}}%
1373       \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
1374       {\csgdef{glo@\glsdetoklabel{##2}@loclist}{}}%
1375       {}%
1376       \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
1377     }%
1378   \else

```

Disable document definitions.

```

1379     \@glxtrdocdeffalse

```

```

1380 \fi
1381 \disable@keys{glossaries-extra}{\docdef}%
1382 }%
1383 {%
1384 \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1385 not permitted\MessageBreak
1386 with record=\@glxtr@record@setting\space package option}%
1387 {You may only use \string\makenoidxglossaries\ space with the
1388 record=off option}%
1389 }%
1390 }

```

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

```

1391 \renewcommand*{\gls@defdocnewglossaryentry}{%
1392 \ifcase\@glxtr@docdefval
1393 docdef=false:
1394 \renewcommand*{\newglossaryentry}[2]{%
1395 \PackageError{glossaries-extra}{Glossary entries must
1396 be \MessageBreak defined in the preamble with \MessageBreak
1397 package option ‘docdef=false’\MessageBreak(consider using
1398 ‘docdef=restricted’)}{Move your glossary definitions to
1399 the preamble. You can also put them in a \MessageBreak separate file
1400 and load them with \string\loadglsentries.}%
1401 }%
1402 \or
1403 docdef=true Since the see value is now saved in a field, it can be used by entries that have
1404 been defined in the document.
1405 \let\gls@checkseeallowed\relax
1406 \let\newglossaryentry\newglossaryentry
1407 \or
1408 Restricted mode just needs to allow the see value.
1409 \let\gls@checkseeallowed\relax
1410 \fi
1411 }%

```

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`

```

1408 \newcommand*{\GlsXtrEnableOnTheFly}{%
1409 \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1410 }

```

`\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.)

```

1411 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1412   \renewcommand*{\glsdetoklabel}[1]{%
1413     \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1414     {%
1415       \expandafter\detokenize\expandafter{##1}%
1416     }%
1417     {\detokenize{##1}}}%
1418   }%
1419   \@GlsXtrEnableOnTheFly
1420 }
1421 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1422   \expandafter\if\glsbackslash#1%
1423     #3%
1424   \else
1425     #4%
1426   \fi
1427 }

```

sxtrstarflywarn

```

1428 \newcommand*{\glsxtrstarflywarn}{%
1429   \GlossariesExtraWarning{Experimental starred version of
1430   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1431   read the warnings in the glossaries-extra user manual)}}%
1432 }

```

rEnableOnTheFly

```

1433 \newcommand*{\@GlsXtrEnableOnTheFly}{%

```

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

These definitions are all assigned the category given by:

`\glsxtrcat`

```

1434 \newcommand*{\glsxtrcat}{general}

```

`\glsxtr`

```

1435 \newcommand*{\glsxtr}[1] [] {%
1436   \def\glsxtr@keylist{##1}%
1437   \@glsxtr
1438 }

```

`\@glsxtr`

```

1439 \newcommand*{\@glsxtr}[2] [] {%
1440   \ifglstryexists{##2}%
1441   {%
1442     \ifblank{##1}{ }\{\@GlsXtrWarning{##1}{##2}}%

```

```

1443 }%
1444 {%
1445   \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1446     description={\nopostdesc},##1}%
1447 }%
1448 \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
1449 }

```

\Glsxtr

```

1450 \newcommand*\Glsxtr}[1][]{%
1451   \def\glsxtr@keylist{##1}%
1452   \@Glsxtr
1453 }

```

\@Glsxtr

```

1454 \newcommand*\@Glsxtr}[2][]{%
1455   \ifglsentryexists{##2}%
1456   {%
1457     \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
1458   }%
1459   {%
1460     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1461       description={\nopostdesc},##1}%
1462   }%
1463   \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
1464 }

```

\glsxtrpl

```

1465 \newcommand*\glsxtrpl}[1][]{%
1466   \def\glsxtr@keylist{##1}%
1467   \@glsxtrpl
1468 }

```

\@glsxtrpl

```

1469 \newcommand*\@glsxtrpl}[2][]{%
1470   \ifglsentryexists{##2}%
1471   {%
1472     \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
1473   }%
1474   {%
1475     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1476       description={\nopostdesc},##1}%
1477   }%
1478   \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
1479 }

```

\Glsxtrpl

```

1480 \newcommand*\Glsxtrpl}[1][]{%
1481   \def\glsxtr@keylist{##1}%

```



```

1482 \Glsxtrpl
1483 }

```

\Glsxtrpl

```

1484 \newcommand*{\Glsxtrpl}[2][]{%
1485 \ifglstryexists{##2}
1486 {%
1487 \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
1488 }%
1489 {%
1490 \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1491 description={\nopostdesc},##1}%
1492 }%
1493 \expandafter\Glspl\expandafter[\glstr@keylist]{##2}%
1494 }

```

\GlsXtrWarning

```

1495 \newcommand*{\GlsXtrWarning}[2]{%
1496 \def\glstr@optlist{##1}%
1497 \@onelevel@sanitize\glstr@optlist
1498 \GlossariesExtraWarning{The options '\glstr@optlist' have
1499 been ignored for entry '##2' as it has already been defined}%
1500 }

```

Disable commands after the glossary:

```

1501 \renewcommand\@printglossary[2]{%
1502 \def\glstr@printglossopts{##1}%
1503 \@glstr@orgprintglossary{##1}{##2}%
1504 \def\glstr{\glstr@disabledflycommand\glstr}%
1505 \def\glstrpl{\glstr@disabledflycommand\glstrpl}%
1506 \def\Glsxtr{\glstr@disabledflycommand\Glsxtr}%
1507 \def\Glsxtrpl{\glstr@disabledflycommand\Glsxtrpl}%
1508 }

```

abledflycommand

```

1509 \newcommand*{\glstr@disabledflycommand}[1]{%
1510 \PackageError{glossaries-extra}%
1511 {\string##1\space can't be used after any of the \MessageBreak
1512 glossaries have been displayed}%
1513 {The on-the-fly commands enabled by
1514 \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1515 before the glossaries. If you want to use any entries \MessageBreak
1516 after any of the glossaries, you must use the standard \MessageBreak
1517 method of first defining the entry and then using the \MessageBreak
1518 entry with commands like \string\gls}%
1519 @@glstr@disabledflycommand
1520 }%
1521 \newcommand*{@@glstr@disabledflycommand}[2][]{##2}

```

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

```
1522 \let\GlsXtrEnableOnTheFly\relax
1523 }
1524 \@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.

```
1525 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}
```

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

`etglossarystyle`

```
1526 \renewcommand*{\setglossarystyle}[1]{%
1527   \ifcsundef{@glsstyle@#1}%
1528   {%
1529     \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
1530   }%
1531   {%
1532     \csname @glsstyle@#1\endcsname
```

Only set the current style if it exists.

```
1533   \protected@edef\@glxtr@current@style{#1}%
1534   }%
1535   \ifx\@glossary@default@style\relax
1536     \protected@edef\@glossary@default@style{#1}%
1537   \fi
1538 }
```

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

```
1539 \ifdef\@glossary@default@style
1540 {}
1541 {%
1542   \let\@glossary@default@style\relax
1543 }
```

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

```
1544 \ifdef\glslistdottedwidth
1545 {%
1546   \ifdim\glslistdottedwidth=.5\hsize
1547     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
1548   \AtBeginDocument{%
1549     \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
1550       \setlength{\glslistdottedwidth}{.5\columnwidth}%
1551     \fi
```

```

1552     }%
1553   \fi
1554 }
1555 {}%

```

Similarly for \glsdescwidth:

\glsdescwidth

```

1556 \ifdef\glsdescwidth
1557 {%
1558   \ifdim\glsdescwidth=.6\hsize
1559     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
1560     \AtBeginDocument{%
1561       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
1562         \setlength{\glsdescwidth}{.6\columnwidth}%
1563     }
1564   }%
1565 \fi
1566 }
1567 {}%

```

and for \glspagelistwidth:

\glspagelistwidth

```

1568 \ifdef\glspagelistwidth
1569 {%
1570   \ifdim\glspagelistwidth=.1\hsize
1571     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
1572     \AtBeginDocument{%
1573       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
1574         \setlength{\glspagelistwidth}{.1\columnwidth}%
1575     }
1576   }%
1577 \fi
1578 }
1579 {}%

```

aryentrynumbers Has the nonnumberlist option been used?

```

1580 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
1581 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
1582   \glsnonnumberlistfalse
1583   \renewcommand*{\glossaryentrynumbers}[1]{%
1584     \ifglstryexists{\glscurrententrylabel}%
1585     {%
1586       \@glsxtrpreloctag
1587       \GlsXtrFormatLocationList{#1}%
1588       \@glsxtrpostloctag
1589       \gls@save@numberlist{#1}%
1590     }{}%
1591   }%

```

```

1592 \else
1593   \glsnonnumberlisttrue
1594   \renewcommand*{\glossaryentrynumbers}[1]{%
1595     \ifglsentryexists{\glscurrententrylabel}%
1596     {%
1597       \gls@save@numberlist{#1}%
1598     }{}%
1599   }%
1600 \fi

```

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

```

1601 \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`

```

1602 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
1603   \let\@glxtrpreloctag\@glxtrpreloctag
1604   \let\@glxtrpostloctag\@glxtrpostloctag
1605   \renewcommand*{\@glxtr@pagetag}{#1}%
1606   \renewcommand*{\@glxtr@pagetag}{#2}%
1607   \renewcommand*{\@glxtr@savepreloctag}[2]{%
1608     \csgdef{\@glxtr@preloctag##1}{##2}%
1609   }%
1610   \renewcommand*{\@glxtr@doloctag}{%
1611     \ifcsundef{\@glxtr@preloctag\glscurrententrylabel}%
1612     {%
1613       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
1614         Rerun required}%
1615     }%
1616     {%
1617       \csuse{\@glxtr@preloctag\glscurrententrylabel}%
1618     }%
1619   }%
1620 }
1621 \@onlypreamble\GlsXtrEnablePreLocationTag

```

`\glxtrpreloctag`

```

1622 \newcommand*{\@glxtrpreloctag}{%
1623   \let\@glxtr@org@delimN\delimN
1624   \let\@glxtr@org@delimR\delimR
1625   \let\@glxtr@org@glsgignore\glsgignore

```

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

```

1626 \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1627 \renewcommand*{\delimN}{\%
1628 \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1629 \@glsxtr@org@delimN}%
1630 \renewcommand*{\delimR}{\%
1631 \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1632 \@glsxtr@org@delimR}%
1633 \renewcommand*{\glsignore}[1]{\%
1634 \gdef\@glsxtr@thisloctag{\relax}%
1635 \@glsxtr@org@glsignore{##1}}%
1636 \@glsxtr@doloctag
1637 }

```

glsxtrpreloctag

```
1638 \newcommand*{\@glsxtrpreloctag}{}
```

@glsxtr@pagetag

```
1639 \newcommand*{\@glsxtr@pagetag}{}%
```

glsxtr@pagetag

```
1640 \newcommand*{\@glsxtr@pagetag}{}%
```

lsxtrpostloctag

```

1641 \newcommand*{\@glsxtrpostloctag}{\%
1642 \let\delimN\@glsxtr@org@delimN
1643 \let\delimR\@glsxtr@org@delimR
1644 \let\glsignore\@glsxtr@org@glsignore
1645 \protected@write\@auxout{\%
1646 {\string\@glsxtr@savepreloctag{\glscurrententrylabel}\@glsxtr@thisloctag}}%
1647 }

```

lsxtrpostloctag

```
1648 \newcommand*{\@glsxtrpostloctag}{}
```

lsxtr@preloctag

```

1649 \newcommand*{\@glsxtr@savepreloctag}[2]{\%
1650 \protected@write\@auxout{\%
1651 \string\providecommand\string\@glsxtr@savepreloctag[2]{}}

```

glsxtr@doloctag

```
1652 \newcommand*{\@glsxtr@doloctag}{}
```

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

```

1653 \renewcommand*{\KV@printgloss@nonumberlist}[1]{\%
1654 \XKV@plfalse
1655 \XKV@sttrue
1656 \XKV@checkchoice[\XKV@resa]{#1}{true,false}%

```

```

1657 {%
1658   \csname glsnonumberlist\XKV@resa\endcsname
1659   \ifglsnonumberlist
1660     \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
1661   \else
1662     \def\glossaryentrynumbers##1{%
1663       \@glstrpreloctag
1664       \GlsXtrFormatLocationList{##1}%
1665       \@glstrpostloctag
1666       \gls@save@numberlist{##1}}%
1667   \fi
1668 }%
1669 }

```

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.

```

1670 \renewcommand*{\glsentryfmt}{%
1671   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}%
1672   \glsifregular{\glslabel}%
1673   {\glstrregularfont{\glsgenentryfmt}}%
1674   {%
1675     \ifglshasshort{\glslabel}%
1676     {\glstrgenabbrvfmt}%
1677     {\glstrregularfont{\glsgenentryfmt}}%
1678   }%
1679 }

```

sxtrregularfont Font used for regular entries.

```
1680 \newcommand*{\glstrregularfont}[1]{#1}
```

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

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

```
1681 \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).

```

1682 \@glxtr@record{#2}{#3}{glslink}%
1683 \glsdoifexists{#3}%
1684 {%

```

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).

```

1685 \let\glxtrorg@ifKV@glslink@hyper@ifKV@glslink@hyper
1686 \let\do@gls@link@checkfirsthyper@gls@link@nocheckfirsthyper
1687 \def\glscustomtext{#4}%
1688 \@glxtr@field@linkdefs
1689 #1%
1690 \@gls@link[#2]{#3}{#4}%
1691 \let@ifKV@glslink@hyper\glxtrorg@ifKV@glslink@hyper
1692 }%
1693 \glspostlinkhook
1694 }

```

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

\@gls@ Save the original definition and redefine.

```

1695 \let\@glxtr@org@gls@\@gls@
1696 \def\@gls@#1#2{%
1697 \@glxtr@record{#1}{#2}{glslink}%
1698 \@glxtr@org@gls@{#1}{#2}%
1699 }%

```

\@glspl@ Save the original definition and redefine.

```

1700 \let\@glxtr@org@glspl@\@glspl@
1701 \def\@glspl@#1#2{%
1702 \@glxtr@record{#1}{#2}{glslink}%
1703 \@glxtr@org@glspl@{#1}{#2}%
1704 }%

```

\@Gls@ Save the original definition and redefine.

```

1705 \let\@glxtr@org@Gls@\@Gls@
1706 \def\@Gls@#1#2{%
1707 \@glxtr@record{#1}{#2}{glslink}%
1708 \@glxtr@org@Gls@{#1}{#2}%
1709 }%

```

\@Glspl@ Save the original definition and redefine.

```

1710 \let\@glxtr@org@Glspl@\@Glspl@
1711 \def\@Glspl@#1#2{%
1712 \@glxtr@record{#1}{#2}{glslink}%
1713 \@glxtr@org@Glspl@{#1}{#2}%
1714 }%

```

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

```
1715 \let\@glxtr@org@GLS@\@GLS@
1716 \def\@GLS@#1#2{%
1717   \@glxtr@record{#1}{#2}{glslink}%
1718   \@glxtr@org@GLS@{#1}{#2}%
1719 }%
```

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

```
1720 \let\@glxtr@org@GLSpl@\@GLSpl@
1721 \def\@GLSpl@#1#2{%
1722   \@glxtr@record{#1}{#2}{glslink}%
1723   \@glxtr@org@GLSpl@{#1}{#2}%
1724 }%
```

`\@glsdisp` Save the original definition and redefine. Can't save and restore `\@glsdisp` since it has an optional argument.

```
1725 \renewcommand*\@glsdisp}[3][{}]{%
1726   \@glxtr@record{#1}{#2}{glslink}%
1727   \glsdoifexists{#2}{%
1728     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
1729     \let\glsifplural\@secondoftwo
1730     \let\glscapscase\@firstofthree
1731     \def\glscustomtext{#3}%
1732     \def\glsinsert{}%
1733     \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
1734     \@gl@link[#1]{#2}{\@glo@text}%
1735     \ifKV@glslink@local
1736       \glslocalunset{#2}%
1737     \else
1738       \glsunset{#2}%
1739     \fi
1740   }%
1741   \glspostlinkhook
1742 }
```

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

```
1743 \renewcommand*\@gls@link}[3][{}]{%
1744   \@glxtr@record{#1}{#2}{glslink}%
1745   \glsdoifexistsordo{#2}%
1746   {%
1747     \let\do@gl@link@checkfirsthyper\relax
1748     \@gl@link[#1]{#2}{#3}%
1749   }%
1750   {%
1751     \glstextformat{#3}%
1752   }%
1753   \glspostlinkhook
1754 }
```


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

```
1755 \newcommand*{\glsxtrinitwrgloss}{%
1756   \glsifattribute{\glslabel}{wrgloss}{after}%
1757   {%
1758     \glsxtrinitwrglossbeforefalse
1759   }%
1760   {%
1761     \glsxtrinitwrglossbeforetrue
1762   }%
1763 }
```

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

```
1764 \newif\ifglsxtrinitwrglossbefore
1765 \glsxtrinitwrglossbeforetrue
```

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

```
1766 \define@choicekey{glslink}{wrgloss}[\val\nr]{before,after}%
1767 {%
1768   \ifcase\nr\relax
1769     \glsxtrinitwrglossbeforetrue
1770   \or
1771     \glsxtrinitwrglossbeforefalse
1772   \fi
1773 }
```

```
1774 \define@key{glslink}{thevalue}{\def\@glsxtr@thevalue{#1}}
```

```
1775 \define@key{glslink}{theHvalue}{\def\@glsxtr@theHvalue{#1}}
```

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

```
1776 \define@boolkey{glslink}[glsxtr@]{hyperoutside}[true]{}
1777 \glsxtr@hyperoutsidetrue
```

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

```
1778 \newcommand*{\glsxtrinithyperoutside}{%
1779   \glsifattribute{\glslabel}{hyperoutside}{false}%
1780   {%
1781     \glsxtr@hyperoutsidefalse
1782   }%
1783   {%
1784     \glsxtr@hyperoutsidetrue
1785   }%
1786 }
```

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

```
1787 \newcommand*{\glsxtr@inc@linkcount}{}%
```

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

```
1788 \newcommand*{\glslinkpresetkeys}{}
```

`\@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`.

```
1789 \def\@gls@link[#1]#2#3{%
1790   \leavevmode
1791   \edef\glslabel{\glsdetoklabel{#2}}%
1792   \def\@gls@link@opts{#1}%
1793   \let\@gls@link@label\glslabel
1794   \let\@gls@numberformat\@glsxtr@defaultnumberformat
1795   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
1796   \edef\@gls@type{\csname glo@\glslabel @type\endcsname}%
1797   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Initialise the value and the H value (v1.19).

```
1798   \def\@glsxtr@thevalue{%
1799   \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

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

```
1800   \glsxtr@initwrgloss
```

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

```
1801   \glsxtr@inithyperoutside
```

Note that the default link options may override `\glsxtr@initwrgloss`.

```
1802   \@gls@setdefault@glslink@opts
```

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

```
1803   \glsxtr@inc@linkcount
```

As the original definition.

```
1804   \do@gls@disablehyperinlist
1805   \do@gls@link@checkfirsthyper
```

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

```
1806   \glslinkpresetkeys
```

Set options.

```
1807   \setkeys{glslink}{#1}%
```

User hook after options are set:

```
1808   \glslinkpostsetkeys
```

Check the value and the H value before saving (v1.19).

```
1809   \ifdefempty{\@glsxtr@thevalue}%
1810   {%
1811     \@gls@saveentrycounter
1812   }%
1813   {%
1814     \let\theglsentrycounter\@glsxtr@thevalue
1815     \def\theHglentrycounter{\@glsxtr@theHvalue}%
```

```

1816 }%
1817 \@gls@setsort{\glslabel}%
    Check textformat attribute (new to v1.21).
1818 \gls@hasattribute{\glslabel}{textformat}%
1819 {%
1820   \edef\@glsxtr@attrval{\gls@getattribute{\glslabel}{textformat}}%
1821   \ifcsdef{\@glsxtr@attrval}%
1822   {%
1823     \letcs{\@glsxtr@textformat}{\@glsxtr@attrval}%
1824   }%
1825   {%
1826     \GlossariesExtraWarning{Unknown control sequence name
1827       '\@glsxtr@attrval' supplied in textformat attribute
1828       for entry '\glslabel'. Reverting to default \string\glstextformat}%
1829     \let\@glsxtr@textformat\glstextformat
1830   }%
1831 }%
1832 {%
1833   \let\@glsxtr@textformat\glstextformat
1834 }%

```

Do write if it should occur before the link text:

```

1835 \ifglsxtrinitwrglossbefore
1836   \@do@wrglossary{#2}%
1837 \fi

```

Do the link text:

```

1838 \ifKV@glslink@hyper
1839   \ifglsxtr@hyperoutside
1840     \@glslink{\glslinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
1841   \else
1842     \@glsxtr@textformat{\@glslink{\glslinkprefix\glslabel}{#3}}%
1843   \fi
1844 \else
1845   \ifglsxtr@hyperoutside
1846     \glsdonohyperlink{\glslinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
1847   \else
1848     \@glsxtr@textformat{\glsdonohyperlink{\glslinkprefix\glslabel}{#3}}%
1849   \fi
1850 \fi

```

Do write if it should occur after the link text:

```

1851 \ifglsxtrinitwrglossbefore
1852   \else
1853     \@do@wrglossary{#2}%
1854   \fi

```

As the original definition:

```

1855 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
1856 }

```

```

1857 \define@key{glossadd}{thevalue}{\def\@glxtr@thevalue{#1}}
1858 \define@key{glossadd}{theHvalue}{\def\@glxtr@theHvalue{#1}}

```

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

```

1859 \renewrobustcmd*{\glsadd}[2][\%
1860   \glxtrifinmark
1861   {}%
1862   {%
1863     \@gls@adjustmode
1864     \@glxtr@record{#1}{#2}{glossadd}%
1865     \glsdoifexists{#2}%
1866     {%
1867       \let\@glsnumberformat\@glxtr@defaultnumberformat
1868       \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
1869       \def\@glxtr@thevalue{}%
1870       \def\@glxtr@theHvalue{\@glxtr@thevalue}%
1871       \setkeys{glossadd}{#1}%
1872       \ifdefempty{\@glxtr@thevalue}%
1873       {%
1874         \@gls@saveentrycounter
1875       }%
1876       {%
1877         \let\theglentrycounter\@glxtr@thevalue
1878         \def\theHglentrycounter{\@glxtr@theHvalue}%
1879       }%

```

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

```

1880     \@gls@setsort{#2}%
1881     \@do@wrglossary{#2}%
1882   }%
1883 }%
1884 }

```

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

```

1885 \newcommand*{\@glxtr@field@linkdefs}{%
1886   \let\glxtrifwasfirstuse\@secondoftwo
1887   \let\glsifplural\@secondoftwo
1888   \let\glscapscase\@firstofthree
1889   \let\glsinsert\@empty
1890 }

```

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

`assignfieldfont`

```

1891 \newcommand*{\glxtrassignfieldfont}[1]{%
1892   \ifglentryexists{#1}%
1893   {%
1894     \ifglshasshort{#1}%

```

```

1895   {%
1896     \glssetabbrvfmt{\glscategory{#1}}%
1897     \glsifregular{#1}%
1898     {\let\@gls@field@font\glxtrregularfont}%
1899     {\let\@gls@field@font\@firstofone}%
1900   }%
1901   {%
1902     \glsifnotregular{#1}%
1903     {\let\@gls@field@font\@firstofone}%
1904     {\let\@gls@field@font\glxtrregularfont}%
1905   }%
1906 }%
1907 {%
1908   \let\@gls@field@font\@gobble
1909 }%
1910 }

```

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

```

1911 \def\@glstext@#1#2[#3]{%
1912   \glxtrassignfieldfont{#2}%
1913   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
1914 }

```

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

```

1915 \def\@GLStext@#1#2[#3]{%
1916   \glxtrassignfieldfont{#2}%
1917   \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
1918   {\@gls@field@font{\@GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
1919 }

```

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

```

1920 \def\@Glstext@#1#2[#3]{%
1921   \glxtrassignfieldfont{#2}%
1922   \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
1923   {\@gls@field@font{\@GLSaccesstext{#2}#3}}%
1924 }

```

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`

```

1925 \newcommand*{\glxtrchecknohyperfirst}[1]{%
1926   \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
1927 }

```

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

```

1928 \def\@glsfirst@#1#2[#3]{%
1929   \glxtrassignfieldfont{#2}%

```

Ensure that \glsfirst honours the nohyperfirst attribute.

```
1930 \@gls@field@link
1931 [\let\glsxtrifwasfirstuse\@firstoftwo
1932 \glsxtrchecknohyperfirst{#2}%
1933 ]{#1}{#2}%
1934 {\@gls@field@font{\glsaccessfirst{#2}#3}}%
1935 }
```

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

```
1936 \def\@Glsfirst@#1#2[#3]{%
1937 \glsxtrassignfieldfont{#2}%
1938 \@gls@field@link
1939 [\let\glsxtrifwasfirstuse\@firstoftwo
1940 \let\gls@scapscase\@secondofthree
1941 \glsxtrchecknohyperfirst{#2}%
1942 ]%
1943 {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
1944 }
```

Ensure that \Glsfirst honours the nohyperfirst attribute.

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

```
1945 \def\@GLSfirst@#1#2[#3]{%
1946 \glsxtrassignfieldfont{#2}%
1947 \@gls@field@link
1948 [\let\glsxtrifwasfirstuse\@firstoftwo
1949 \let\gls@scapscase\@thirdofthree
1950 \glsxtrchecknohyperfirst{#2}%
1951 ]%
1952 {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
1953 }
```

Ensure that \GLSfirst honours the nohyperfirst attribute.

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

```
1954 \def\@glsplural@#1#2[#3]{%
1955 \glsxtrassignfieldfont{#2}%
1956 \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
1957 {\@gls@field@font{\glsaccessplural{#2}#3}}%
1958 }
```

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

```
1959 \def\@Glsplural@#1#2[#3]{%
1960 \glsxtrassignfieldfont{#2}%
1961 \@gls@field@link
1962 [\let\glsifplural\@firstoftwo
1963 \let\gls@scapscase\@secondofthree
1964 ]%
1965 {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
1966 }
```

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

```
1967 \def\@GLSplural@#1#2[#3]{%
1968   \glstrassignfieldfont{#2}%
1969   \@gls@field@link
1970   [\let\glsifplural\@firstoftwo
1971   \let\glscapscase\@thirdofthree
1972   ]%
1973   {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
1974 }
```

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

```
1975 \def\@glsfirstplural@#1#2[#3]{%
1976   \glstrassignfieldfont{#2}%
1977   \@gls@field@link
1978   [\let\glstrifwasfirstuse\@firstoftwo
1979   \let\glsifplural\@firstoftwo
1980   \glstrchecknohyperfirst{#2}%
1981   ]%
1982   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
1983 }
```

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

```
1984 \def\@GLsfirstplural@#1#2[#3]{%
1985   \glstrassignfieldfont{#2}%
1986   \@gls@field@link
1987   [\let\glstrifwasfirstuse\@firstoftwo
1988   \let\glsifplural\@firstoftwo
1989   \let\glscapscase\@secondofthree
1990   \glstrchecknohyperfirst{#2}%
1991   ]%
1992   {#1}{#2}{\@gls@field@font{\GLsaccessfirstplural{#2}#3}}%
1993 }
```

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

```
1994 \def\@GLSfirstplural@#1#2[#3]{%
1995   \glstrassignfieldfont{#2}%
1996   \@gls@field@link
1997   [\let\glstrifwasfirstuse\@firstoftwo
1998   \let\glsifplural\@firstoftwo
1999   \let\glscapscase\@thirdofthree
2000   \glstrchecknohyperfirst{#2}%
2001   ]%
2002   {#1}{#2}%
2003   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2004 }
```

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

```
2005 \def\@glsname@#1#2[#3]{%
2006   \glsxtrassignfieldfont{#2}%
2007   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}#3}}%
2008 }
```

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

```
2009 \def\@Glsname@#1#2[#3]{%
2010   \glsxtrassignfieldfont{#2}%
2011   \@gls@field@link
2012   [\let\gls@scapscase\@secondoftwo]{#1}{#2}%
2013   {\@gls@field@font{\Glsaccessname{#2}#3}}%
2014 }
```

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

```
2015 \def\@GLSname@#1#2[#3]{%
2016   \glsxtrassignfieldfont{#2}%
2017   \@gls@field@link[\let\gls@scapscase\@thirdoftwo]%
2018   {#1}{#2}%
2019   {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2020 }
```

\@glsdesc@

```
2021 \def\@glsdesc@#1#2[#3]{%
2022   \glsxtrassignfieldfont{#2}%
2023   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
2024 }
```

\@Glsdesc@ First letter uppercase version.

```
2025 \def\@Glsdesc@#1#2[#3]{%
2026   \glsxtrassignfieldfont{#2}%
2027   \@gls@field@link
2028   [\let\gls@scapscase\@secondoftwo]{#1}{#2}%
2029   {\@gls@field@font{\Glsaccessdesc{#2}#3}}%
2030 }
```

\@GLSdesc@ All uppercase version.

```
2031 \def\@GLSdesc@#1#2[#3]{%
2032   \glsxtrassignfieldfont{#2}%
2033   \@gls@field@link[\let\gls@scapscase\@thirdoftwo]%
2034   {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2035 }
```

@glsdescplural@ No case-changing version.

```
2036 \def\@glsdescplural@#1#2[#3]{%
2037   \glsxtrassignfieldfont{#2}%
2038   \@gls@field@link
2039   [\let\gls@scapscase\@secondoftwo
```



```

2040 \let\glsifplural\@firstoftwo
2041 ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
2042 }

```

@Glsdescplural@ First letter uppercase version.

```

2043 \def\@Glsdescplural@#1#2[#3]{%
2044 \glsxtrassignfieldfont{#2}%
2045 \@gls@field@link
2046 [\let\glscapscase\@secondoftwo
2047 \let\glsifplural\@firstoftwo
2048 ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
2049 }

```

@GLSdescplural@ All uppercase version.

```

2050 \def\@GLSdesc@#1#2[#3]{%
2051 \glsxtrassignfieldfont{#2}%
2052 \@gls@field@link
2053 [\let\glscapscase\@thirdoftwo
2054 \let\glsifplural\@firstoftwo
2055 ]%
2056 {#1}{#2}%
2057 {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
2058 }

```

\@glssymbol@

```

2059 \def\@glssymbol@#1#2[#3]{%
2060 \glsxtrassignfieldfont{#2}%
2061 \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
2062 }

```

\@Glsymbol@ First letter uppercase version.

```

2063 \def\@Glsymbol@#1#2[#3]{%
2064 \glsxtrassignfieldfont{#2}%
2065 \@gls@field@link
2066 [\let\glscapscase\@secondoftwo]%
2067 {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
2068 }

```

\@GLSsymbol@ All uppercase version.

```

2069 \def\@GLSsymbol@#1#2[#3]{%
2070 \glsxtrassignfieldfont{#2}%
2071 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2072 {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
2073 }

```

lssymbolplural@ No case-changing version.

```

2074 \def\@glssymbolplural@#1#2[#3]{%
2075 \glsxtrassignfieldfont{#2}%

```

```

2076 \@gls@field@link
2077 [\let\glscapscase\@secondoftwo
2078 \let\glsifplural\@firstoftwo
2079 ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2080 }

```

lssymbolplural@ First letter uppercase version.

```

2081 \def\@GLssymbolplural@#1#2[#3]{%
2082 \glstrassignfieldfont{#2}%
2083 \@gls@field@link
2084 [\let\glscapscase\@secondoftwo
2085 \let\glsifplural\@firstoftwo
2086 ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
2087 }

```

LSsymbolplural@ All uppercase version.

```

2088 \def\@GLSsymbol@#1#2[#3]{%
2089 \glstrassignfieldfont{#2}%
2090 \@gls@field@link
2091 [\let\glscapscase\@thirdoftwo
2092 \let\glsifplural\@firstoftwo
2093 ]%
2094 {#1}{#2}%
2095 {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2096 }

```

\@Glsuseri@ First letter uppercase version.

```

2097 \def\@Glsuseri@#1#2[#3]{%
2098 \glstrassignfieldfont{#2}%
2099 \@gls@field@link
2100 [\let\glscapscase\@secondoftwo]{#1}{#2}%
2101 {\@gls@field@font{\Glsentryuseri{#2}#3}}%
2102 }

```

\@GLSuseri@ All uppercase version.

```

2103 \def\@GLSuseri@#1#2[#3]{%
2104 \glstrassignfieldfont{#2}%
2105 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2106 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}}%
2107 }

```

\@Glsuserii@ First letter uppercase version.

```

2108 \def\@Glsuserii@#1#2[#3]{%
2109 \glstrassignfieldfont{#2}%
2110 \@gls@field@link
2111 [\let\glscapscase\@secondoftwo]%
2112 {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
2113 }

```

\@GLSuserii@ All uppercase version.

```
2114 \def\@GLSuserii@#1#2[#3]{%
2115   \glstrassignfieldfont{#2}%
2116   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2117   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2118 }
```

\@GLSuseriii@ First letter uppercase version.

```
2119 \def\@GLSuseriii@#1#2[#3]{%
2120   \glstrassignfieldfont{#2}%
2121   \@gls@field@link
2122   [\let\glscapscase\@secondoftwo]%
2123   {#1}{#2}{\@gls@field@font{\Glentryuseriii{#2}#3}}}%
2124 }
```

\@GLSuseriii@ All uppercase version.

```
2125 \def\@GLSuseriii@#1#2[#3]{%
2126   \glstrassignfieldfont{#2}%
2127   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2128   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2129 }
```

\@GLSuseriv@ First letter uppercase version.

```
2130 \def\@GLSuseriv@#1#2[#3]{%
2131   \glstrassignfieldfont{#2}%
2132   \@gls@field@link
2133   [\let\glscapscase\@secondoftwo]%
2134   {#1}{#2}{\@gls@field@font{\Glentryuseriv{#2}#3}}}%
2135 }
```

\@GLSuseriv@ All uppercase version.

```
2136 \def\@GLSuseriv@#1#2[#3]{%
2137   \glstrassignfieldfont{#2}%
2138   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2139   {#1}{#2}%
2140   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2141 }
```

\@GLSuserv@ First letter uppercase version.

```
2142 \def\@GLSuserv@#1#2[#3]{%
2143   \glstrassignfieldfont{#2}%
2144   \@gls@field@link
2145   [\let\glscapscase\@secondoftwo]%
2146   {#1}{#2}{\@gls@field@font{\Glentryuserv{#2}#3}}}%
2147 }
```

\@GLSuserv@ All uppercase version.

```
2148 \def\@GLSuserv@#1#2[#3]{%
```

```

2149 \glstrassignfieldfont{#2}%
2150 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2151 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2152 }

```

\@Glsuservi@ First letter uppercase version.

```

2153 \def\@Glsuservi@#1#2[#3]{%
2154 \glstrassignfieldfont{#2}%
2155 \@gls@field@link
2156 [\let\glscapscase\@secondoftwo]%
2157 {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}}%
2158 }

```

\@GLSuservi@ All uppercase version.

```

2159 \def\@GLSuservi@#1#2[#3]{%
2160 \glstrassignfieldfont{#2}%
2161 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2162 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
2163 }

```

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

\@acrshort No case change.

```

2164 \def\@acrshort#1#2[#3]{%
2165 \glsdoifexists{#2}%
2166 {%
2167 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2168 \let\glxtrifwasfirstuse\@secondoftwo
2169 \let\glsifplural\@secondoftwo
2170 \let\glscapscase\@firstofthree
2171 \let\glsinsert\@empty
2172 \def\glscustomtext{%
2173 \acronymfont{\glsaccesssshort{#2}}#3}%
2174 }%
2175 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2176 }%
2177 \glspostlinkhook
2178 }

```

\@Acrshort First letter uppercase.

```

2179 \def\@Acrshort#1#2[#3]{%
2180 \glsdoifexists{#2}%
2181 {%
2182 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2183 \let\glxtrifwasfirstuse\@secondoftwo
2184 \let\glsifplural\@secondoftwo
2185 \let\glscapscase\@secondofthree
2186 \let\glsinsert\@empty

```

```

2187 \def\glscustomtext{%
2188 \acronymfont{\Glsaccesssshort{#2}}#3%
2189 }%
2190 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2191 }%
2192 \glspostlinkhook
2193 }

```

\@ACRshort All uppercase.

```

2194 \def\@ACRshort#1#2[#3]{%
2195 \glsdofexists{#2}%
2196 {%
2197 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2198 \let\glstrifwasfirstuse\@secondoftwo
2199 \let\glsifplural\@secondoftwo
2200 \let\glscapscase\@thirdofthree
2201 \let\glsinsert\@empty
2202 \def\glscustomtext{%
2203 \mfirstucMakeUppercase{\acronymfont{\Glsaccesssshort{#2}}#3}%
2204 }%
2205 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2206 }%
2207 \glspostlinkhook
2208 }

```

\@acrshortpl No case change.

```

2209 \def\@acrshortpl#1#2[#3]{%
2210 \glsdofexists{#2}%
2211 {%
2212 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2213 \let\glstrifwasfirstuse\@secondoftwo
2214 \let\glsifplural\@firstoftwo
2215 \let\glscapscase\@firstofthree
2216 \let\glsinsert\@empty
2217 \def\glscustomtext{%
2218 \acronymfont{\Glsaccesssshortpl{#2}}#3%
2219 }%
2220 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2221 }%
2222 \glspostlinkhook
2223 }

```

\@Acrshortpl First letter uppercase.

```

2224 \def\@Acrshortpl#1#2[#3]{%
2225 \glsdofexists{#2}%
2226 {%
2227 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2228 \let\glstrifwasfirstuse\@secondoftwo
2229 \let\glsifplural\@firstoftwo

```

```

2230 \let\glscapscase\@secondofthree
2231 \let\glsinsert\@empty
2232 \def\glscustomtext{%
2233     \acronymfont{\Glsaccessshortpl{#2}}#3%
2234 }%
2235 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2236 }%
2237 \glspostlinkhook
2238 }

```

\@ACRshortpl All uppercase.

```

2239 \def\@ACRshortpl#1#2[#3]{%
2240     \glsdoifexists{#2}%
2241     {%
2242         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2243         \let\glstrifwasfirstuse\@secondoftwo
2244         \let\glsifplural\@firstoftwo
2245         \let\glscapscase\@thirdofthree
2246         \let\glsinsert\@empty
2247         \def\glscustomtext{%
2248             \mfirstucMakeUppercase{\acronymfont{\Glsaccessshortpl{#2}}#3}%
2249         }%
2250         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2251     }%
2252     \glspostlinkhook
2253 }

```

\@acrlong No case change.

```

2254 \def\@acrlong#1#2[#3]{%
2255     \glsdoifexists{#2}%
2256     {%
2257         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2258         \let\glstrifwasfirstuse\@secondoftwo
2259         \let\glsifplural\@secondoftwo
2260         \let\glscapscase\@firstofthree
2261         \let\glsinsert\@empty
2262         \def\glscustomtext{%
2263             \acronymfont{\Glsaccesslong{#2}}#3%
2264         }%
2265         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2266     }%
2267     \glspostlinkhook
2268 }

```

\@Acrlong First letter uppercase.

```

2269 \def\@Acrlong#1#2[#3]{%
2270     \glsdoifexists{#2}%
2271     {%
2272         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

2273 \let\glxtrifwasfirstuse\@secondoftwo
2274 \let\glsifplural\@secondoftwo
2275 \let\glscapscase\@secondofthree
2276 \let\glsinsert\@empty
2277 \def\glscustomtext{%
2278   \acronymfont{\Glsaccesslong{#2}}#3%
2279 }%
2280 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2281 }%
2282 \glspostlinkhook
2283 }

```

\@ACRlong All uppercase.

```

2284 \def\@ACRlong#1#2[#3]{%
2285   \glsdoifexists{#2}%
2286   {%
2287     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2288     \let\glxtrifwasfirstuse\@secondoftwo
2289     \let\glsifplural\@secondoftwo
2290     \let\glscapscase\@thirdofthree
2291     \let\glsinsert\@empty
2292     \def\glscustomtext{%
2293       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslong{#2}}#3}%
2294     }%
2295     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2296   }%
2297   \glspostlinkhook
2298 }

```

\@acrlongpl No case change.

```

2299 \def\@acrlongpl#1#2[#3]{%
2300   \glsdoifexists{#2}%
2301   {%
2302     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2303     \let\glxtrifwasfirstuse\@secondoftwo
2304     \let\glsifplural\@firstoftwo
2305     \let\glscapscase\@firstofthree
2306     \let\glsinsert\@empty
2307     \def\glscustomtext{%
2308       \acronymfont{\Glsaccesslongpl{#2}}#3%
2309     }%
2310     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2311   }%
2312   \glspostlinkhook
2313 }

```

\@Acrlongpl First letter uppercase.

```

2314 \def\@Acrlongpl#1#2[#3]{%
2315   \glsdoifexists{#2}%

```

```

2316 {%
2317   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2318   \let\glxtrifwasfirstuse\@secondoftwo
2319   \let\gl@sifplural\@firstoftwo
2320   \let\glscapscase\@secondofthree
2321   \let\gl$insert\@empty
2322   \def\glscustomtext{%
2323     \acronymfont{\Glsaccesslongpl{#2}}#3%
2324   }%
2325   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2326 }%
2327 \glspostlinkhook
2328 }

```

\@ACRlongpl All uppercase.

```

2329 \def\@ACRlongpl#1#2[#3]{%
2330   \gl@dofexists{#2}%
2331   {%
2332     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2333     \let\glxtrifwasfirstuse\@secondoftwo
2334     \let\gl@sifplural\@firstoftwo
2335     \let\glscapscase\@thirdofthree
2336     \let\gl$insert\@empty
2337     \def\glscustomtext{%
2338       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslongpl{#2}}#3}%
2339     }%
2340     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2341   }%
2342   \glspostlinkhook
2343 }

```

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

\@glsaddkey

```

2344 \renewcommand*{\@glsaddkey}[7]{%
2345   \key@ifundefined{glossentry}{#1}%
2346   {%
2347     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2348     \appto\@gl@keymap{, {#1}{#1}}%
2349     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
2350     \appto\@newglossaryentryposthook{%
2351       \letcs{\@glo@tmp}{@glo@#1}%
2352       \gl@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2353     }%
2354     \newcommand*{#3}[1]{\@gl@entry@field{##1}{#1}}%
2355     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2356   \ifcsdef{@gl@user@#1@}%

```



```

2357 {%
2358   \PackageError{glossaries}%
2359   {Can't define '\string#5' as helper command
2360   '\expandafter\string\csname @gls@user@#1@\endcsname' already
2361   exists}%
2362   {}}%
2363 }%
2364 {%
2365   \expandafter\newcommand\expandafter*\expandafter
2366   {\csname @gls@user@#1@\endcsname}[2][{}]{%
2367     \new@ifnextchar[%
2368       {\csuse{@gls@user@#1@}{##1}{##2}}%
2369       {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
2370   \csdef{@gls@user@#1@}##1##2[##3]{%
2371     \@gls@field@link{##1}{##2}{#3{##2}##3}%
2372   }%
2373   \newrobustcmd*{#5}{%
2374     \expandafter\@gls@hyp@opt\csname @gls@user@#1@\endcsname}%
2375   }%

```

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

```

2376   \ifcsdef{@Gls@user@#1@}%
2377   {%
2378     \PackageError{glossaries}%
2379     {Can't define '\string#6' as helper command
2380     '\expandafter\string\csname @Gls@user@#1@\endcsname' already
2381     exists}%
2382     {}}%
2383 }%
2384 {%
2385   \expandafter\newcommand\expandafter*\expandafter
2386   {\csname @Gls@user@#1@\endcsname}[2][{}]{%
2387     \new@ifnextchar[%
2388       {\csuse{@Gls@user@#1@}{##1}{##2}}%
2389       {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
2390   \csdef{@Gls@user@#1@}##1##2[##3]{%
2391     \@gls@field@link[\let\glscaps@case\@secondofthree]%
2392     {##1}{##2}{#4{##2}##3}%
2393   }%
2394   \newrobustcmd*{#6}{%
2395     \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
2396   }%

```

Finally the all caps version (modified):

```

2397   \ifcsdef{@GLS@user@#1@}%
2398   {%
2399     \PackageError{glossaries}%
2400     {Can't define '\string#7' as helper command
2401     '\expandafter\string\csname @GLS@user@#1@\endcsname' already
2402     exists}%

```

```

2403     {}%
2404 }%
2405 {%
2406     \expandafter\newcommand\expandafter*\expandafter
2407     {\csname @GLS@user@#1\endcsname}[2][{}]{%
2408         \new@ifnextchar[%
2409             {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2410             {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
2411     \csdef{@GLS@user@#1@}##1##2[##3]{%
2412         \@gls@field@link[\let\gls@caps@case\@thirdofthree]%
2413         {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2414     }%
2415     \newrobustcmd*{#7}{%
2416         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2417     }%
2418 }%
2419 {%
2420     \PackageError{glossaries-extra}{Key ‘#1’ already exists}{}%
2421 }%
2422 }

```

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

```

2423 \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.

```

2424 \let\@gls@xtr@org@checkfirsthyper\@gls@link@checkfirsthyper
2425 \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.

```

2426 \ifglsused{\glslabel}%
2427 {\let\gls@xtrifwasfirstuse\@secondoftwo}
2428 {\let\gls@xtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```

2429 \edef\gls@categorylabel{\gls@category{\glslabel}}%
2430 \ifglsused{\glslabel}%
2431 {%
2432     \glsifcategoryattribute{\gls@categorylabel}{nohypernext}{true}%
2433     {\KV@glslink@hyperfalse}}%
2434 }%
2435 {%
2436     \glsifcategoryattribute{\gls@categorylabel}{nohyperfirst}{true}%
2437     {\KV@glslink@hyperfalse}}%
2438 }%
2439 \glslinkcheckfirsthyperhook
2440 }

```

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.

```
2441 \ifdef\do@gl:disablehyperinlist
2442 {%
2443   \let\@gl:extr@do@gl:disablehyperinlist\do@gl:disablehyperinlist
2444   \renewcommand*{\do@gl:disablehyperinlist}{%
2445     \@gl:extr@do@gl:disablehyperinlist
2446     \gl:sifattribute{\gl:label}{nohyper}{true}{\KV@gl:link@hyperfalse}}}%
2447   }
2448 }
2449 }
```

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

```
2450 \define@boolkey{gl:link}{noindex}[true]{}
2451 \KV@gl:link@noindexfalse
```

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

lt@gl:link@opts

```
2452 \ifdef\@gl:link@setdefault@gl:link@opts
2453 {
2454   \renewcommand*{\@gl:link@setdefault@gl:link@opts}{%
2455     \KV@gl:link@noindexfalse
2456     \@gl:extrasetaliasnoindex
2457   }
2458 }
2459 {
```

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

```
2460   \newcommand*{\@gl:link@setdefault@gl:link@opts}{%
2461     \KV@gl:link@noindexfalse
2462     \@gl:extrasetaliasnoindex
2463   }
2464   \preto\do@gl:disablehyperinlist{\@gl:link@setdefault@gl:link@opts}
2465 }
```

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.)

```
2466 \providecommand*{\gl:extrasetaliasnoindex}{%
2467   \KV@gl:link@noindextrue
2468 }
```

setaliasnoindex

```
2469 \newcommand*{\@gl:extrasetaliasnoindex}{%
2470   \gl:extrifhasfield{alias}{\gl:label}%
2471   {%
2472     \let\gl:extrindexaliased\@gl:extrindexaliased
```

```

2473 \glxtrsetaliasnoindex
2474 \let\glxtrindexaliased\@no@glxtrindexaliased
2475 }%
2476 {}%
2477 }

```

xtrindexaliased

```

2478 \newcommand{\@glxtrindexaliased}{%
2479 \ifKV@glslink@noindex
2480 \else
2481 \begingroup
2482 \let\@glslnumberformat\@glxtr@defaultnumberformat
2483 \edef\@glscounter{\csname glo@glsdetoklabel{\glslabel}@counter\endcsname}%
2484 \glxtr@saveentrycounter
2485 \@do@wrglossary{\glxtralias{\glslabel}}%
2486 \endgroup
2487 \fi
2488 }

```

xtrindexaliased

```

2489 \newcommand{\@no@glxtrindexaliased}{%
2490 \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
2491 not permitted outside definition of \string\glxtrsetaliasnoindex}%
2492 {}%
2493 }

```

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

```

2494 \let\glxtrindexaliased\@no@glxtrindexaliased

```

tDefaultGlsOpts Set the default options for \glslink etc.

```

2495 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
2496 \renewcommand*{\@glscsetdefault@glslink@opts}{%
2497 \setkeys{glslink}{#1}%
2498 \@glxtrsetaliasnoindex
2499 }%
2500 }

```

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

```

2501 \newcommand*{\glxtrifindexing}[2]{%
2502 \ifKV@glslink@noindex #2\else #1\fi
2503 }

```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```

2504 \renewcommand*{\glswriteentry}[2]{%
2505 \glxtrifindexing
2506 {%
2507 \ifglindexonlyfirst
2508 \ifglused{#1}

```

```

2509     {\glxtrdoautoindexname{#1}{dualindex}}}%
2510     {#2}}%
2511   \else
2512     \glusifattribute{#1}{indexonlyfirst}{true}%
2513     {\ifglused{#1}
2514       {\glxtrdoautoindexname{#1}{dualindex}}}%
2515       {#2}}}%
2516     {#2}}%
2517   \fi
2518 }%
2519 {}%
2520 }

```

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

```

2521 \appto\@do@@wrglossary{\@glxtr@do@@wrindex
2522   \glxtrdowrglossaryhook{\@gls@label}}%
2523 }

```

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

Similarly for the “noidx” version:

`s@noidxglossary`

```

2524 \appto\gls@noidxglossary{\@glxtr@do@@wrindex
2525   \glxtrdowrglossaryhook{\@gls@label}}%
2526 }

```

`xtr@do@@wrindex`

```

2527 \newcommand*{\@glxtr@do@@wrindex}{%
2528   \glxtrdoautoindexname{\@gls@label}{dualindex}}%
2529 }

```

`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.)

```

2530 \newcommand*{\glxtrdowrglossaryhook}[1]{%

```

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

```

2531 \newcommand*{\@gls@alt@hyp@opt}[1]{%
2532   \let\glslinkvar\@firstofthree
2533   \let\@gls@hyp@opt@cs#1\relax
2534   \@ifstar{\s@gls@hyp@opt}%
2535   {\@ifnextchar+%
2536     {\@firstoftwo{\p@gls@hyp@opt}}}%
2537   {%
2538     \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
2539     {\@firstoftwo{\@alt@gls@hyp@opt}}}%
2540   {#1}}%

```

```

2541 }%
2542 }%
2543 }

alt@gls@hyp@opt User version
2544 \newcommand*{\@alt@gls@hyp@opt}[1] [] {%
2545 \let\glslinkvar\@firstofthree
2546 \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}

lt@hyp@opt@char Contains the character used as the command modifier.
2547 \newcommand*{\@gls@alt@hyp@opt@char}{}

lt@hyp@opt@keys Contains the option list used as the command modifier.
2548 \newcommand*{\@gls@alt@hyp@opt@keys}{}

rSetAltModifier
2549 \newcommand*{\GlsXtrSetAltModifier}[2] {%
2550 \let\@gls@hyp@opt\@gls@alt@hyp@opt
2551 \def\@gls@alt@hyp@opt@char{#1}%
2552 \def\@gls@alt@hyp@opt@keys{#2}%
2553 }

org@dohyperlink
2554 \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.
2555 \ifdef\glsnavhyperlink
2556 {
2557 \renewcommand*{\glsnavhyperlink}[3] [\@glo@type] {%
2558 \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
Scope:
2559 {%
2560 \let\glsdohyperlink\glsxtr@org@dohyperlink
2561 \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
2562 }%
2563 }%
2564 }
2565 {}

\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 tem-
porarily makes \gls behave like \glstext[hyper=false,noindex]. (This will be overrid-
den if the user explicitly cancels either of those options in the optional argument of \gls
or using the plus version.) This also patches the short form commands like \acrshort

```

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

```

2566 \renewcommand*{\glsdohyperlink}[2]{%
2567   \glshasattribute{\glslabel}{targeturl}%
2568   {%
2569     \glshasattribute{\glslabel}{targetname}%
2570     {%
2571       \glshasattribute{\glslabel}{targetcategory}%
2572       {%
2573         \hyperref{\glsggetattribute{\glslabel}{targeturl}}{%
2574           {\glsggetattribute{\glslabel}{targetcategory}}}%
2575         {\glsggetattribute{\glslabel}{targetname}}}%
2576         {\glxtrprotectlinks#2}}%
2577       }%
2578     {%
2579       \hyperref{\glsggetattribute{\glslabel}{targeturl}}{%
2580         }%
2581         {\glsggetattribute{\glslabel}{targetname}}}%
2582         {\glxtrprotectlinks#2}}%
2583     }%
2584   }%
2585   {%
2586     \href{\glsggetattribute{\glslabel}{targeturl}}{%
2587       {\glxtrprotectlinks#2}}%
2588     }%
2589   }%
2590   {%

```

Check for alias.

```

2591   \glsgfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
2592   \ifdefvoid\gloaliaslabel
2593   {%
2594     \glxtrhyperlink{#1}{\glxtrprotectlinks#2}}%
2595   }%
2596   {%

```

Redirect link to the alias target.

```

2597   \glxtrhyperlink
2598   {\glolinkprefix\glsgdetoklabel{\gloaliaslabel}}%
2599   {\glxtrprotectlinks#2}}%
2600   }%
2601 }%
2602 }

```

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

```

2603 \ifdef\@glsshowtarget
2604 {
2605   \newcommand{\glxtrhyperlink}[2]{%
2606     \@glsshowtarget{#1}%
2607     \hyperlink{#1}{#2}%

```

```

2608 }%
2609 }
2610 {
2611   \newcommand{\glstrhyperlink}[2]{\hyperlink{#1}{#2}}%
2612 }

```

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

```

2613 \renewrobustcmd*{\glshyperlink}[2][\glstrytext{\@glo@label}]{%
2614   \glsdoifexists{#2}%
2615   {%
2616     \def\@glo@label{#2}%
2617     {\edef\glslabel{#2}%
2618       \@glslink{\glo@linkprefix\glslabel}{#1}}%
2619   }%
2620 }

```

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

```

2621 \renewcommand{\glsdisablehyper}{%
2622   \KV@glslink@hyperfalse
2623   \def\@glslink{\glsdohyperlink}%
2624   \let\@glstarget\@secondoftwo
2625 }

```

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

```

2626 \renewcommand{\glsenablehyper}{%
2627   \KV@glslink@hypertrue
2628   \def\@glslink{\glsdohyperlink}%
2629   \def\@glstarget{\glsdohypertarget}%
2630 }

```

`glsdohyperlink` 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.

```

2631 \def\glsdohyperlink#1#2{{\glstrprotectlinks #2}}

```

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

```

2632 \ifcsundef{hyperlink}%
2633 {%
2634   \def\@glslink{\glsdohyperlink}
2635 }%
2636 {%
2637   \def\@glslink{\glsdohyperlink}
2638 }

```


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

```

2639 \newcommand*{\glsxtrprotectlinks}{%
2640   \KV@glslink@hyperfalse
2641   \KV@glslink@noindextrue
2642   \let\@gls@\@glsxtr@p@text@
2643   \let\@Gls@\@Glsxtr@p@text@
2644   \let\@GLS@\@GLSxtr@p@text@
2645   \let\@glspl@\@glsxtr@p@plural@
2646   \let\@Glspl@\@Glsxtr@p@plural@
2647   \let\@GLSpl@\@GLSxtr@p@plural@
2648   \let\@glsxtrshort@\@glsxtr@p@short@
2649   \let\@Glsxtrshort@\@Glsxtr@p@short@
2650   \let\@GLSxtrshort@\@GLSxtr@p@short@
2651   \let\@glsxtrlong@\@glsxtr@p@long@
2652   \let\@Glsxtrlong@\@Glsxtr@p@long@
2653   \let\@GLSxtrlong@\@GLSxtr@p@long@
2654   \let\@glsxtrshortpl@\@glsxtr@p@shortpl@
2655   \let\@Glsxtrshortpl@\@Glsxtr@p@shortpl@
2656   \let\@GLSxtrshortpl@\@GLSxtr@p@shortpl@
2657   \let\@glsxtrlongpl@\@glsxtr@p@longpl@
2658   \let\@Glsxtrlongpl@\@Glsxtr@p@longpl@
2659   \let\@GLSxtrlongpl@\@GLSxtr@p@longpl@
2660   \let\@acrshort@\@glsxtr@p@acrshort@
2661   \let\@Acrshort@\@Glsxtr@p@acrshort@
2662   \let\@ACRshort@\@GLSxtr@p@acrshort@
2663   \let\@acrshortpl@\@glsxtr@p@acrshortpl@
2664   \let\@Acrshortpl@\@Glsxtr@p@acrshortpl@
2665   \let\@ACRshortpl@\@GLSxtr@p@acrshortpl@
2666   \let\@acrlong@\@glsxtr@p@acrlong@
2667   \let\@Acrlong@\@Glsxtr@p@acrlong@
2668   \let\@ACRlong@\@GLSxtr@p@acrlong@
2669   \let\@acrlongpl@\@glsxtr@p@acrlongpl@
2670   \let\@Acrlongpl@\@Glsxtr@p@acrlongpl@
2671   \let\@ACRlongpl@\@GLSxtr@p@acrlongpl@
2672 }

```

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

@glsxtr@p@text@

```

2673 \def\@glsxtr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}

```

@Glsxtr@p@text@

```

2674 \def\@Glsxtr@p@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}

```

@GLSxtr@p@text@

```

2675 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}

```

lsxtr@p@plural@

```

2676 \def\@glsxtr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}

```

lsxtr@p@plural@

```
2677 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}
```

LSxtr@p@plural@

```
2678 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}
```

glxtr@p@short@

```
2679 \def\@glxtr@p@short@#1#2[#3]{%
2680 {%
2681   \glsssetabbrvfmt{\glscategory{#2}}%
2682   \glsabbrvfont{\glentryshort{#2}}#3%
2683 }%
2684 }
```

Glsxtr@p@short@

```
2685 \def\@Glsxtr@p@short@#1#2[#3]{%
2686 {%
2687   \glsssetabbrvfmt{\glscategory{#2}}%
2688   \glsabbrvfont{\Glsentryshort{#2}}#3%
2689 }%
2690 }
```

GLSxtr@p@short@

```
2691 \def\@GLSxtr@p@short@#1#2[#3]{%
2692 {%
2693   \glsssetabbrvfmt{\glscategory{#2}}%
2694   \mfirstucMakeUppercase{\glsabbrvfont{\glentryshort{#2}}#3}%
2695 }%
2696 }
```

sxtr@p@shortpl@

```
2697 \def\@glxtr@p@shortpl@#1#2[#3]{%
2698 {%
2699   \glsssetabbrvfmt{\glscategory{#2}}%
2700   \glsabbrvfont{\glentryshortpl{#2}}#3%
2701 }%
2702 }
```

Sxtr@p@shortpl@

```
2703 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
2704 {%
2705   \glsssetabbrvfmt{\glscategory{#2}}%
2706   \glsabbrvfont{\Glsentryshortpl{#2}}#3%
2707 }%
2708 }
```

GSxtr@p@shortpl@

```
2709 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
```

```

2710  {%
2711    \glsssetabbrvfmt{\glscategory{#2}}%
2712    \mfirstucMakeUppercase{\glssabbrvfont{\glssentryshortpl{#2}}#3}%
2713  }%
2714 }

```

@glssxtr@p@long@

```
2715 \def\@glssxtr@p@long@#1#2[#3]{\glssentrylong{#2}#3}}
```

@Glsxtr@p@long@

```
2716 \def\@Glsxtr@p@long@#1#2[#3]{\Glsentrylong{#2}#3}}
```

@GLSxtr@p@long@

```

2717 \def\@GLSxtr@p@long@#1#2[#3]{%
2718   {\mfirstucMakeUppercase{\glsslongfont{\glssentrylong{#2}}#3}}}

```

lssxtr@p@longpl@

```
2719 \def\@glssxtr@p@longpl@#1#2[#3]{\glssentrylongpl{#2}#3}}
```

Lssxtr@p@longpl@

```
2720 \def\@Lssxtr@p@longpl@#1#2[#3]{\glsslongfont{\Glsentrylongpl{#2}}#3}}
```

LSxtr@p@longpl@

```

2721 \def\@GLSxtr@p@longpl@#1#2[#3]{%
2722   {\mfirstucMakeUppercase{\glsslongfont{\glssentrylongpl{#2}}#3}}}

```

xtr@p@acrshort@

```
2723 \def\@glssxtr@p@acrshort@#1#2[#3]{\acronymfont{\glssentryshort{#2}}#3}}
```

Xtr@p@acrshort@

```
2724 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}}
```

GLSxtr@p@acrshort@

```

2725 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
2726   {\mfirstucMakeUppercase{\acronymfont{\glssentryshort{#2}}#3}}}

```

glssxtr@p@acrshortpl@

```
2727 \def\@glssxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glssentryshortpl{#2}}#3}}
```

Glsxtr@p@acrshortpl@

```
2728 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}}
```

GLSxtr@p@acrshortpl@

```

2729 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
2730   {\mfirstucMakeUppercase{\acronymfont{\glssentryshortpl{#2}}#3}}}

```

glssxtr@p@acrlong@

```
2731 \def\@glssxtr@p@acrlong@#1#2[#3]{\glssentrylong{#2}#3}}
```

sxtr@p@acrlong@

```
2732 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}
```

Sxtr@p@acrlong@

```
2733 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
```

```
2734 {\mfirstucMakeUppercase{\Glsentrylong{#2}#3}}}
```

tr@p@acrlongpl@

```
2735 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
2736 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
2737 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
```

```
2738 {\mfirstucMakeUppercase{\Glsentrylongpl{#2}#3}}}
```

Commands to minimise conflict.

\@glsxtrp@opt

```
2739 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}
```

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

```
2740 \newcommand*{\glsxtrsetpopts}[1]{%
```

```
2741 \renewcommand*{\@glsxtrp@opt}{#1}%
```

```
2742 }
```

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

```
2743 \newcommand*{\glossxtrsetpopts}{%
```

```
2744 \glsxtrsetpopts{noindex}%
```

```
2745 }
```

\@@glsxtrp

```
2746 \newrobustcmd*{\@@glsxtrp}[2]{%
```

Add scope.

```
2747 {%
```

```
2748 \let\glspostlinkhook\relax
```

```
2749 \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
```

```
2750 }%
```

```
2751 }
```

\@glsxtrp

```
2752 \newrobustcmd*{\@glsxtrp}[2]{%
```

```
2753 \ifcsdef{gls#1}%
```

```
2754 {%
```

```
2755 \@glsxtrp{gls#1}{#2}%
```

```
2756 }%
```

```

2757 {%
2758   \ifcsdef{glsxtr#1}%
2759   {%
2760     \@glsxtrp{glsxtr#1}{#2}%
2761   }%
2762   {%
2763     \PackageError{glossaries-extra}{‘#1’ not recognised by
2764       \string\glsxtrp}{}%
2765   }%
2766 }%
2767 }

```

\@Glsxtrp

```

2768 \newrobustcmd*{\@Glsxtrp}[2]{%
2769   \ifcsdef{Gls#1}%
2770   {%
2771     \@glsxtrp{Gls#1}{#2}%
2772   }%
2773   {%
2774     \ifcsdef{Glsxtr#1}%
2775     {%
2776       \@glsxtrp{Glsxtr#1}{#2}%
2777     }%
2778     {%
2779       \PackageError{glossaries-extra}{‘#1’ not recognised by
2780         \string\Glsxtrp}{}%
2781     }%
2782   }%
2783 }

```

\@GLSxtrp

```

2784 \newrobustcmd*{\@GLSxtrp}[2]{%
2785   \ifcsdef{GLS#1}%
2786   {%
2787     \@glsxtrp{GLS#1}{#2}%
2788   }%
2789   {%
2790     \ifcsdef{GLSxtr#1}%
2791     {%
2792       \@glsxtrp{GLSxtr#1}{#2}%
2793     }%
2794     {%
2795       \PackageError{glossaries-extra}{‘#1’ not recognised by
2796         \string\GLSxtrp}{}%
2797     }%
2798   }%
2799 }

```

\glsxtr@entry@p

```

2800 \newrobustcmd*{\glsxtr@headentry@p}[2]{%
2801   \glsifattribute{#1}{headuc}{true}%
2802   {%
2803     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
2804   }%
2805   {%
2806     \@gls@entry@field{#1}{#2}%
2807   }%
2808 }

```

`\glsxtrp` Not robust as it needs to expand somewhat.

```

2809 \ifdef\teorpdfstring
2810 {
2811   \newcommand{\glsxtrp}[2]{%
2812     \protect\NoCaseChange
2813     {%
2814       \protect\teorpdfstring
2815       {%
2816         \protect\glsxtrifinmark
2817         {%
2818           \ifcsdef{glsxtrhead#1}%
2819           {%
2820             {\protect\csuse{glsxtrhead#1}{#2}}%
2821           }%
2822           {%
2823             \glsxtr@headentry@p{#2}{#1}%
2824           }%
2825         }%
2826         {%
2827           \@glsxtrp{#1}{#2}%
2828         }%
2829       }%
2830     }%
2831     \protect\@gls@entry@field{#2}{#1}%
2832   }%
2833 }%
2834 }
2835 }
2836 {
2837   \newcommand{\glsxtrp}[2]{%
2838     \protect\NoCaseChange
2839     {%
2840       \protect\glsxtrifinmark
2841       {%
2842         \ifcsdef{glsxtrhead#1}%
2843         {%
2844           {\protect\csuse{glsxtrhead#1}}%
2845         }%
2846         {%

```

```

2847         \glxtr@headentry@p{#2}{#1}%
2848     }%
2849 }%
2850 {%
2851     \@glxtrp{#1}{#2}%
2852 }%
2853 }%
2854 }
2855 }

```

Provide short synonyms for the most common option.

\glsp

```

2856 \newcommand*{\glsp}{\glxtrp{short}}

```

\glsp

```

2857 \newcommand*{\glsp}{\glxtrp{text}}

```

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

```

2858 \ifdef\texorpdfstring
2859 {
2860     \newcommand{\Glsxtrp}[2]{%
2861         \protect\NoCaseChange
2862         {%
2863             \protect\texorpdfstring
2864             {%
2865                 \protect\glxtrifinmark
2866                 {%
2867                     \ifcsdef{Glsxtrhead#1}%
2868                     {%
2869                         {\protect\csuse{Glsxtrhead#1}{#2}}%
2870                     }%
2871                     {%
2872                         \protect\@Gls@entry@field{#2}{#1}%
2873                     }%
2874                 }%
2875                 {%
2876                     \@Glsxtrp{#1}{#2}%
2877                 }%
2878             }%
2879             {%
2880                 \protect\@gl@entry@field{#2}{#1}%
2881             }%
2882         }%
2883     }
2884 }
2885 {
2886     \newcommand{\Glsxtrp}[2]{%

```

```

2887 \protect\NoCaseChange
2888 {%
2889 \protect\glxtrifinmark
2890 {%
2891 \ifcsdef{Glsxtrhead#1}%
2892 {%
2893 {\protect\csuse{Glsxtrhead#1}}%
2894 }%
2895 {%
2896 \protect\@Gls@entry@field{#2}{#1}%
2897 }%
2898 }%
2899 {%
2900 \@Glsxtrp{#1}{#2}%
2901 }%
2902 }%
2903 }
2904 }

```

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

```

2905 \ifdef\teorpdfstring
2906 {
2907 \newcommand{\GLSxtrp}[2]{%
2908 \protect\NoCaseChange
2909 {%
2910 \protect\teorpdfstring
2911 {%
2912 \protect\glxtrifinmark
2913 {%
2914 \ifcsdef{GLSxtr#1}%
2915 {%
2916 {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
2917 }%
2918 {%
2919 \protect\mfirstucMakeUppercase
2920 {%
2921 \protect\@gls@entry@field{#2}{#1}%
2922 }%
2923 }%
2924 }%
2925 {%
2926 \@GLSxtrp{#1}{#2}%
2927 }%
2928 }%
2929 {%
2930 \protect\@gls@entry@field{#2}{#1}%
2931 }%
2932 }%
2933 }

```



```

2934 }
2935 {
2936   \newcommand{\GLSxtrp}[2]{%
2937     \protect\NoCaseChange
2938     {%
2939       \protect\glxtrifinmark
2940       {%
2941         \ifcsdef{GLSxtr#1}%
2942         {%
2943           {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
2944         }%
2945       }%
2946       \protect\mfirstucMakeUppercase
2947       {%
2948         \protect\@gls@entry@field{#2}{#1}%
2949       }%
2950     }%
2951   }%
2952   {%
2953     \@GLSxtrp{#1}{#2}%
2954   }%
2955 }%
2956 }
2957 }

```

1.3.5 Entry Counting

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

First adjust definitions of the unset and reset commands to provide a hook.

\@glsunset Global unset.

```

2958 \renewcommand*{\@glsunset}[1]{%
2959   \@glsunset{#1}%
2960   \glxtrpostunset{#1}%
2961 }%

```

glxtrpostunset

```

2962 \newcommand*{\glxtrpostunset}[1]{%

```

\@glslocalunset Local unset.

```

2963   \renewcommand*{\@glslocalunset}[1]{%
2964     \@glslocalunset{#1}%
2965     \glxtrpostlocalunset{#1}%

```

```

2966 }%

rpostlocalunset
2967 \newcommand*{\glxtrpostlocalunset}[1]{}

\@glsreset Global reset.
2968 \renewcommand*{\@glsreset}[1]{%
2969 \@@glsreset{#1}%
2970 \glxtrpostreset{#1}%
2971 }%

glxtrpostreset
2972 \newcommand*{\glxtrpostreset}[1]{}

\@glslocalreset Local reset.
2973 \renewcommand*{\@glslocalreset}[1]{%
2974 \@@glslocalreset{#1}%
2975 \glxtrpostlocalreset{#1}%
2976 }%

rpostlocalreset
2977 \newcommand*{\glxtrpostlocalreset}[1]{}

leEntryCounting The first argument is the list of categories and the second argument is the value of the en-
trycount attribute.
2978 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
Enable entry counting:
2979 \glsenableentrycount
Redefine \gls etc:
2980 \renewcommand*{\gls}{\cglss}%
2981 \renewcommand*{\Gls}{\cGls}%
2982 \renewcommand*{\glspl}{\cglsspl}%
2983 \renewcommand*{\Glspl}{\cGlspl}%
2984 \renewcommand*{\GLS}{\cGLS}%
2985 \renewcommand*{\GLSpl}{\cGLSpl}%
Set the entrycount attribute:
2986 \@glxtr@setentrycountunsetattr{#1}{#2}%
In case this command is used again:
2987 \let\GlsXtrEnableEntryCounting\@glxtr@setentrycountunsetattr
2988 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
2989 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
2990 can't be used with \string\GlsXtrEnableEntryCounting}%
2991 {Use one or other but not both commands}}%
2992 }

```

countunsetattr

```
2993 \newcommand*{\@glstr@setentrycountunsetattr}[2]{%
2994 \@for\@glstr@cat:=#1\do
2995 {%
2996   \ifdefempty{\@glstr@cat}{}%
2997   {%
2998     \glsssetcategoryattribute{\@glstr@cat}{entrycount}{#2}%
2999   }%
3000 }%
3001 }
```

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

enableentrycount

```
3002 \renewcommand*{\glsenableentrycount}{%
  Enable new fields:
3003   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
  Just in case the user has switched on the docdef option.
3004   \renewcommand*{\gls@defdocnewglossaryentry}{%
3005     \renewcommand*{\newglossaryentry}[2]{%
3006       \PackageError{glossaries}{\string\newglossaryentry\space
3007         may only be used in the preamble when entry counting has
3008         been activated}{If you use \string\glsenableentrycount\space
3009         you must place all entry definitions in the preamble not in
3010         the document environment}%
3011     }%
3012   }%
  New commands to access new fields:
3013   \newcommand*{\glscurrentcount}[1]{%
3014     \ifcsundef{glo@\glscdetoklabel{##1}@currentcount}%
3015     {0}{\@gls@entry@field{##1}{currentcount}}%
3016   }%
3017   \newcommand*{\glsprevcount}[1]{%
3018     \ifcsundef{glo@\glscdetoklabel{##1}@prevcount}%
3019     {0}{\@gls@entry@field{##1}{prevcount}}%
3020   }%
  Adjust post unset and reset:
3021   \let\@glstr@entrycount@org@unset\glstrpostunset
3022   \renewcommand*{\glstrpostunset}[1]{%
3023     \@glstr@entrycount@org@unset{##1}%
3024     \@gls@increment@currentcount{##1}%
3025   }%
3026   \let\@glstr@entrycount@org@localunset\glstrpostlocalunset
3027   \renewcommand*{\glstrpostlocalunset}[1]{%
3028     \@glstr@entrycount@org@localunset{##1}%
3029     \@gls@local@increment@currentcount{##1}%
3030   }%
```

```

3031 \let\@glxtr@entrycount@org@reset\glxtrpostreset
3032 \renewcommand*{\glxtrpostreset}[1]{%
3033   \@glxtr@entrycount@org@reset{##1}%
3034   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
3035 }%
3036 \let\@glxtr@entrycount@org@localreset\glxtrpostlocalreset
3037 \renewcommand*{\glxtrpostlocalreset}[1]{%
3038   \@glxtr@entrycount@org@localreset{##1}%
3039   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
3040 }%

```

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

```

3041 \let\@cgl@\@cgl@
3042 \let\@cgl@pl@\@cgl@pl@

3043 \let\@cgl@\@cgl@
3044 \let\@cgl@pl@\@cgl@pl@
3045 \let\@cgl@S@\@cgl@S@
3046 \let\@cgl@Spl@\@cgl@Spl@

```

The rest is as the original definition.

```

3047 \AtEndDocument{\@gl@write@entrycounts}%
3048 \renewcommand*{\@gl@entry@count}[2]{%
3049   \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
3050 }%
3051 \let\gl@enableentrycount\relax
3052 \renewcommand*{\gl@enableentryunitcount}{%
3053   \PackageError{glossaries-extra}{\string\gl@enableentryunitcount\space
3054     can't be used with \string\gl@enableentrycount}%
3055   {Use one or other but not both commands}%
3056 }%
3057 }

```

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

```

3058 \renewcommand*{\@gl@write@entrycounts}{%
3059   \immediate\write\@auxout
3060     {\string\providecommand*{\string\@gl@entry@count}[2]{}}%
3061   \count@=0\relax
3062   \forallgl@sentries{\@gl@sentry}{%
3063     \gl@hasattribute{\@gl@sentry}{entrycount}%
3064     {%
3065       \ifgl@sused{\@gl@sentry}%
3066       {%
3067         \immediate\write\@auxout
3068           {\string\@gl@entry@count{\@gl@sentry}{\gl@sentrycurrcount{\@gl@sentry}}}%
3069       }%
3070     }%
3071     \advance\count@ by \@ne

```

```

3072 }%
3073 {}%
3074 }%
3075 \ifnum\count@=0
3076 \GlossariesExtraWarningNoLine{Entry counting has been enabled
3077 \MessageBreak with \string\glsenableentrycount\space but the
3078 \MessageBreak attribute 'entrycount' hasn't
3079 \MessageBreak been assigned to any of the defined
3080 \MessageBreak entries}%
3081 \fi
3082 }

```

trifcounttrigger

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

```

3083 \newcommand*\glxtrifcounttrigger[3]{%
3084 \glshasattribute{#1}{entrycount}%
3085 {%
3086 \ifnum\gl Sentryprevcount{#1}>\gl sgetattribute{#1}{entrycount}\relax
3087 #3%
3088 \else
3089 #2%
3090 \fi
3091 }%
3092 {#3}%
3093 }

```

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

\@@cgl@

```

3094 \def\@@cgl@#1#2[#3]{%
3095 \glxtrifcounttrigger{#2}%
3096 {%
3097 \cgl sformat{#2}{#3}%
3098 \gl sunset{#2}%
3099 }%
3100 {%
3101 \@@cgl@{#1}{#2}[#3]%
3102 }%
3103 }%

```

\@@cgl spl@

```

3104 \def\@@cgl spl@#1#2[#3]{%
3105 \glxtrifcounttrigger{#2}%
3106 {%
3107 \cgl splformat{#2}{#3}%
3108 \gl sunset{#2}%

```

```

3109 }%
3110 {%
3111     \@glsp1@{#1}-{#2}[#3]%
3112 }%
3113 }%

```

\@@cGls@

```

3114 \def\@@cGls@#1#2[#3]{%
3115     \glxtrifcounttrigger{#2}%
3116     {%
3117         \cGlsformat{#2}{#3}%
3118         \glset{#2}%
3119     }%
3120     {%
3121         \@Gls@{#1}-{#2}[#3]%
3122     }%
3123 }%

```

\@@cGlsp1@

```

3124 \def\@@cGlsp1@#1#2[#3]{%
3125     \glxtrifcounttrigger{#2}%
3126     {%
3127         \cGlsp1format{#2}{#3}%
3128         \glset{#2}%
3129     }%
3130     {%
3131         \@Glsp1@{#1}-{#2}[#3]%
3132     }%
3133 }%

```

\@@cGLS@

```

3134 \def\@@cGLS@#1#2[#3]{%
3135     \glxtrifcounttrigger{#2}%
3136     {%
3137         \cGLSformat{#2}{#3}%
3138         \glset{#2}%
3139     }%
3140     {%
3141         \@GLS@{#1}-{#2}[#3]%
3142     }%
3143 }%

```

\@@cGLSp1@

```

3144 \def\@@cGLSp1@#1#2[#3]{%
3145     \glxtrifcounttrigger{#2}%
3146     {%
3147         \cGLSp1format{#2}{#3}%
3148         \glset{#2}%
3149     }%

```

```

3150  {%
3151    \@GLSp1@{#1}{#2}[#3]%
3152  }%
3153 }%

```

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

```

\@cgl's@
3154 \def\@cgl's@#1#2[#3]{\@gls@{#1}{#2}[#3]}

```

```

\@cGls@
3155 \def\@cGls@#1#2[#3]{\@Gls@{#1}{#2}[#3]}

```

```

\@cgl'spl@
3156 \def\@cgl'spl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}

```

```

\@cGlspl@
3157 \def\@cGlspl@#1#2[#3]{\@Glspl@{#1}{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

```

\cGLS
3158 \newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}

```

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

```

3159 \newcommand*{\@cGLS}[2][{}]{%
3160   \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[]}]%
3161 }

```

```

\@cGLS@
3162 \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.

```

3163 \newcommand*{\cGLSformat}[2]{%
3164   \expandafter\mfirstucMakeUppercase\expandafter{\cgl'sformat{#1}{#2}}%
3165 }

```

```

\cGLSp1
3166 \newrobustcmd*{\cGLSp1}{\@gls@hyp@opt\@cGLSp1}

```

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

```

3167 \newcommand*{\@cGLSp1}[2][{}]{%
3168   \new@ifnextchar[{\@cGLSp1@{#1}{#2}}{\@cGLSp1@{#1}{#2}[]}]%
3169 }

```

\@cGLSp1@

```
3170 \def\@cGLSp1@#1#2[#3]{\@GLSp1@{#1}{#2}[#3]}
```

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

```
3171 \newcommand*\cGLSplformat}[2]{%
```

```
3172 \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
```

```
3173 }
```

Modify the trigger formats to check for the regular attribute.

\cglformat

```
3174 \renewcommand*\cglformat}[2]{%
```

```
3175 \glsifregular{#1}
```

```
3176 {\glsentryfirst{#1}}%
```

```
3177 {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}\#2%
```

```
3178 }
```

\cGlsformat

```
3179 \renewcommand*\cGlsformat}[2]{%
```

```
3180 \glsifregular{#1}
```

```
3181 {\Glsentryfirst{#1}}%
```

```
3182 {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}\#2%
```

```
3183 }
```

\cglsplformat

```
3184 \renewcommand*\cglsplformat}[2]{%
```

```
3185 \glsifregular{#1}
```

```
3186 {\glsentryfirstplural{#1}}%
```

```
3187 {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}\#2%
```

```
3188 }
```

\cGlsplformat

```
3189 \renewcommand*\cGlsplformat}[2]{%
```

```
3190 \glsifregular{#1}
```

```
3191 {\Glsentryfirstplural{#1}}%
```

```
3192 {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}\#2%
```

```
3193 }
```

New code similar to above for unit counting.

defunitcounters

```
3194 \newcommand*\@newglossaryentry@defunitcounters{%
```

```
3195 \edef\@glo@countunit{\csuse{@glsxtr@categoryattr@{\@glo@category @unitcount}}%
```

```
3196 \ifdefvoid\@glo@countunit
```

```
3197 {}%
```

```
3198 {%
```

```
3199 \@glsxtr@ifunitcounter{\@glo@countunit}%
```



```

3200     {}%
3201     {\expandafter\@glsxtr@addunitcounter\expandafter{\@glo@countunit}}%
3202   }%
3203 }

```

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

```

3204 \newcommand*{\@glsxtr@unitcountlist}{}

```

`@addunitcounter`

```

3205 \newcommand*{\@glsxtr@addunitcounter}[1]{%
3206   \listadd{\@glsxtr@unitcountlist}{#1}%
3207   \ifcsundef{glsxtr@theunit@#1}
3208   {%
3209     \ifcsdef{theH#1}%
3210     {\csdef{glsxtr@theunit@#1}{\csuse{theH#1}}}%
3211     {\csdef{glsxtr@theunit@#1}{\csuse{the#1}}}%
3212   }%
3213   {}%
3214 }

```

`r@ifunitcounter`

```

3215 \newcommand*{\@glsxtr@ifunitcounter}[3]{%
3216   \xifinlist{#1}{\@glsxtr@unitcountlist}{#2}{#3}%
3217 }

```

`urrentunitcount`

```

3218 \newcommand*\@glsxtr@currentunitcount[1]{%
3219   glo@\glsdetoklabel{#1}@currunit@\glsgetattribute{#1}{unitcount}.%
3220   \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
3221 }

```

`viousunitcount`

```

3222 \newcommand*\@glsxtr@previousunitcount[1]{%
3223   glo@\glsdetoklabel{#1}@prevunit@\glsgetattribute{#1}{unitcount}.%
3224   \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
3225 }

```

`t@currunitcount`

```

3226 \newcommand*{\@gls@increment@currunitcount}[1]{%
3227   \glsattribute{#1}{unitcount}%
3228   {%
3229     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3230     \ifcsundef{\@glsxtr@csname}%
3231     {%
3232       \csgdef{\@glsxtr@csname}{1}%
3233       \listcsxadd
3234       {glo@\glsdetoklabel{#1}@unitlist}%
3235       {\glsgetattribute{#1}{unitcount}.%

```

```

3236      \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
3237    }%
3238  }%
3239  {%
3240    \csxdef{\@glxstr@csname}%
3241      {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
3242    }%
3243  }%
3244  {}%
3245 }

```

t@currunitcount

```

3246 \newcommand*{\@glsl@local@increment@currunitcount}[1]{%
3247   \glshasattribute{#1}{unitcount}%
3248   {%
3249     \edef\@glxstr@csname{\@glxstr@currentunitcount{#1}}%
3250     \ifcsundef{\@glxstr@csname}%
3251     {%
3252       \csdef{\@glxstr@csname}{1}%
3253       \listcseadd
3254         {glo@\glsdetoklabel{#1}@unitlist}%
3255         {\glsggetattribute{#1}{unitcount}.%
3256         \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
3257       }%
3258     }%
3259     {%
3260       \csedef{\@glxstr@csname}%
3261         {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
3262       }%
3263     }%
3264   }%
3265 }

```

r@currunitcount

```

3266 \newcommand*{\@glxstr@currunitcount}[2]{%
3267   \ifcsundef
3268     {glo@\glsdetoklabel{#1}@currunit@#2}%
3269     {0}%
3270     {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
3271 }%

```

r@prevunitcount

```

3272 \newcommand*{\@glxstr@prevunitcount}[2]{%
3273   \ifcsundef
3274     {glo@\glsdetoklabel{#1}@prevunit@#2}%
3275     {0}%
3276     {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}%
3277 }%

```

entryunitcount

```
3278 \newcommand*{\glsenableentryunitcount}{%
```

Enable new fields:

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

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

```
3280 \renewcommand*{\gls@defdocnewglossaryentry}{%
```

```
3281 \renewcommand*\newglossaryentry[2]{%
```

```
3282 \PackageError{glossaries}{\string\newglossaryentry\space
```

```
3283 may only be used in the preamble when entry counting has
```

```
3284 been activated}{If you use \string\glsenableentryunitcount\space
```

```
3285 you must place all entry definitions in the preamble not in
```

```
3286 the document environment}%
```

```
3287 }%
```

```
3288 }%
```

New commands to access new fields:

```
3289 \newcommand*{\glsentrycurrcount}[1]{%
```

```
3290 \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
```

```
3291 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
```

```
3292 }%
```

```
3293 \newcommand*{\glsentryprevcount}[1]{%
```

```
3294 \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
```

```
3295 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
```

```
3296 }%
```

Access total count:

```
3297 \newcommand*{\glsentryprevtotalcount}[1]{%
```

```
3298 \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
```

```
3299 {0}%
```

```
3300 {%
```

```
3301 \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
```

```
3302 }%
```

```
3303 }%
```

Access max value:

```
3304 \newcommand*{\glsentryprevmaxcount}[1]{%
```

```
3305 \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
```

```
3306 {0}%
```

```
3307 {%
```

```
3308 \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
```

```
3309 }%
```

```
3310 }%
```

Adjust post unset and reset:

```
3311 \let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
```

```
3312 \renewcommand*{\glsxtrpostunset}[1]{%
```

```
3313 \@glsxtr@entryunitcount@org@unset{##1}%
```

```
3314 \@gls@increment@currunitcount{##1}%
```

```
3315 }%
```

```
3316 \let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
```

```

3317 \renewcommand*{\glxtrpostlocalunset}[1]{%
3318   \@glxtr@entryunitcount@org@localunset{##1}%
3319   \@glx@local@increment@currunitcount{##1}%
3320 }%
3321 \let\@glxtr@entryunitcount@org@reset\glxtrpostreset
3322 \renewcommand*{\glxtrpostreset}[1]{%
3323   \glshasattribute{##1}{unitcount}%
3324   {%
3325     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
3326     \ifcsundef{\@glxtr@csname}%
3327     {}%
3328     {\csgdef{\@glxtr@csname}{0}}%
3329   }%
3330   {}%
3331 }%
3332 \let\@glxtr@entryunitcount@org@localreset\glxtrpostlocalreset
3333 \renewcommand*{\glxtrpostlocalreset}[1]{%
3334   \@glxtr@entryunitcount@org@localreset{##1}%
3335   \glshasattribute{##1}{unitcount}%
3336   {%
3337     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
3338     \ifcsundef{\@glxtr@csname}%
3339     {}%
3340     {\csdef{\@glxtr@csname}{0}}%
3341   }%
3342   {}%
3343 }%

```

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

```

3344 \let\@cgl@{\@cgl@}
3345 \let\@cgl@{\@cgl@}

3346 \let\@cgl@{\@cgl@}
3347 \let\@cgl@{\@cgl@}
3348 \let\@cgl@{\@cgl@}
3349 \let\@cgl@{\@cgl@}

```

Write information to the aux file.

```

3350 \AtEndDocument{\@glx@write@entryunitcounts}%
3351 \renewcommand*{\@glx@entry@unitcount}[3]{%
3352   \csgdef{glo@glx@detoklabel{##1}@prevunit@##3}{##2}%
3353   \ifcsundef{glo@glx@detoklabel{##1}@prevunittotal}%
3354   {\csgdef{glo@glx@detoklabel{##1}@prevunittotal}{##2}}%
3355   {%
3356     \csxdef{glo@glx@detoklabel{##1}@prevunittotal}{
3357       \number\numexpr\csuse{glo@glx@detoklabel{##1}@prevunittotal}+##2}%
3358     }%
3359     \ifcsundef{glo@glx@detoklabel{##1}@prevunitmax}%
3360     {\csgdef{glo@glx@detoklabel{##1}@prevunitmax}{##2}}%

```

```

3361   {%
3362     \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
3363       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
3364     \fi
3365   }%
3366 }%
3367 \let\glsenableentryunitcount\relax
3368 \renewcommand*{\glsenableentrycount}{%
3369   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
3370     can't be used with \string\glsenableentryunitcount}%
3371   {Use one or other but not both commands}%
3372 }%
3373 }
3374 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3375 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

3376 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3377   \immediate\write\@auxout
3378   {\string\@gls@entry@unitcount
3379     {\@glsentry}%
3380     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3381     }%
3382     {#1}}%
3383 }

```

entryunitcounts

```

3384 \newcommand*{\@gls@write@entryunitcounts}{%
3385   \immediate\write\@auxout
3386   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3387   \count@=0\relax
3388   \forallglsentries{\@glsentry}{%
3389     \gls@hasattribute{\@glsentry}{unitcount}%
3390     {%
3391       \ifglsused{\@glsentry}%
3392       {%
3393         \forlistcsloop
3394           {\@gls@write@entryunitcounts@do}%
3395           {glo@\glsdetoklabel{\@glsentry}@unitlist}%
3396       }%
3397     }%
3398     \advance\count@ by \@ne
3399   }%
3400 }%
3401 }%
3402 \ifnum\count@=0
3403   \GlossariesExtraWarningNoLine{Entry counting has been enabled

```

```

3404     \MessageBreak with \string\glsenableentryunitcount\space but the
3405     \MessageBreak attribute ‘unitcount’ hasn’t
3406     \MessageBreak been assigned to any of the defined
3407     \MessageBreak entries}%
3408 \fi
3409 }

```

`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.

```

3410 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
    Enable entry counting:
3411   \glsenableentryunitcount
    Redefine \gls etc:
3412   \renewcommand*{\gls}{\cglsl}%
3413   \renewcommand*{\Gls}{\cGls}%
3414   \renewcommand*{\glspl}{\cglspl}%
3415   \renewcommand*{\Glspl}{\cGlspl}%
3416   \renewcommand*{\GLS}{\cGLS}%
3417   \renewcommand*{\GLSpl}{\cGLSpl}%
    Set the entrycount attribute:
3418   \@glxtr@setentryunitcountunsetattr{#1}{#2}{#3}%
    In case this command is used again:
3419   \let\GlsXtrEnableEntryUnitCounting\@glxtr@setentryunitcountunsetattr
3420   \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3421     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3422       can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
3423     {Use one or other but not both commands}}%
3424 }

```

`countunsetattr`

```

3425 \newcommand*{\@glxtr@setentryunitcountunsetattr}[3]{%
3426   \@for\@glxtr@cat:=#1\do
3427   {%
3428     \ifdefempty{\@glxtr@cat}{}%
3429     {%
3430       \glssetcategoryattribute{\@glxtr@cat}{entrycount}{#2}%
3431       \glssetcategoryattribute{\@glxtr@cat}{unitcount}{#3}%
3432     }%
3433   }%
3434 }

```

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.

`GenericNewAcronym`

```

3435 \renewcommand*{\SetGenericNewAcronym}{%
3436   \let\@Gls@entryname\@Gls@acentryname
3437   \renewcommand{\newacronym}[4][{}]{%
3438     \ifdefempty{\@glsacronymlists}%
3439     {%
3440       \def\@glo@type{\acronymtype}%
3441       \setkeys{glossentry}{##1}%
3442       \DeclareAcronymList{\@glo@type}%
3443     }%
3444     {}%
3445     \glskeylisttok{##1}%
3446     \glslabeltok{##2}%
3447     \glsshorttok{##3}%
3448     \glslongtok{##4}%
3449     \newacronymhook
3450     \protected@edef\@do@newglossaryentry{%
3451       \noexpand\newglossaryentry{\the\glslabeltok}%
3452       {%
3453         type=\acronymtype,%
3454         name={\expandonce{\acronymentry{##2}}},%
3455         sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
3456         text={\the\glsshorttok},%
3457         short={\the\glsshorttok},%
3458         shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3459         long={\the\glslongtok},%
3460         longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3461         category=acronym,%
3462         \GenericAcronymFields,%
3463         \the\glskeylisttok
3464       }%
3465     }%
3466     \@do@newglossaryentry
3467   }%
3468   \renewcommand*{\acrfullfmt}[3]{%
3469     \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
3470   \renewcommand*{\Acrfullfmt}[3]{%
3471     \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
3472   \renewcommand*{\ACRfullfmt}[3]{%
3473     \glslink[##1]{##2}{%
3474       \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
3475   \renewcommand*{\acrfullplfmt}[3]{%
3476     \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
3477   \renewcommand*{\Acrfullplfmt}[3]{%
3478     \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%

```

```

3479 \renewcommand*{\ACRfullplfmt}[3]{%
3480   \glslink{##1}{##2}}{%
3481     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
3482 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
3483 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
3484 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
3485 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
3486 }

```

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:

```

3487 \let\@glxtr@org@setacronymstyle\setacronymstyle
3488 \let\@glxtr@org@newacronymstyle\newacronymstyle

```

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

```

3489 \newcommand*{\MakeAcronymsAbbreviations}{%
3490   \renewcommand*{\newacronym}[4][1]{%
3491     \glxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
3492   }%
3493   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
3494   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
3495   \renewcommand*{\setacronymstyle}[1]{%
3496     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}}
3497     unavailable.
3498     Use \string\setabbreviationstyle\space instead.
3499     The original acronym interface can be restored with
3500     \string\RestoreAcronyms}{}%
3501   }%
3502   \renewcommand*{\newacronymstyle}[1]{%
3503     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
3504     available unless you restore the original acronym interface with
3505     \string\RestoreAcronyms}%
3506     \@glxtr@org@newacronymstyle{##1}%
3507   }%
3508 }

```

Switch acronyms to abbreviations:

```

3509 \MakeAcronymsAbbreviations

```

RestoreAcronyms Restore acronyms to glossaries interface.

```

3510 \newcommand*{\RestoreAcronyms}{%
3511   \SetGenericNewAcronym
3512   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
3513   \renewcommand*{\acronymfont}[1]{##1}%
3514   \let\setacronymstyle\@glxtr@org@setacronymstyle
3515   \let\newacronymstyle\@glxtr@org@newacronymstyle

```


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

```

3516 \renewcommand*\@gls@link@checkfirsthyper{%
3517   \ifglsused{\glslabel}%
3518   {\let\glsxtrifwasfirstuse\@secondoftwo}
3519   {\let\glsxtrifwasfirstuse\@firstoftwo}%
3520   \@glsxtr@org@checkfirsthyper
3521 }
3522 \glssetcategoryattribute{acronym}{regular}{false}%
3523 \setacronymstyle{long-short}%
3524 }

```

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

```

3525 \renewcommand*\glsacspace[1]{%
3526   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
3527   \ifdim\dimen@<\glsacspacemax~\else\space\fi
3528 }

```

`\glsacspacemax` Value used in the above.

```

3529 \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`

```

3530 \newcommand*\@glsxtr@reg@glosslist{}

```

Save the original definition of `\makeglossaries`:

```

3531 \let\@glsxtr@org@makeglossaries\makeglossaries

```

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

`\makeglossaries`

```

3532 \renewcommand*\makeglossaries[1][]{%
3533   \ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
3534     \PackageError{glossaries-extra}{\string\makeglossaries\space
3535       not permitted\MessageBreak with record=only package option}%
3536     {You may only use \string\makeglossaries\space with
3537       record=off or record=alsoindex options}%
3538   \else
3539     \ifblank{#1}%
3540     {\@glsxtr@org@makeglossaries}%

```

```

3541 {%
3542   \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
3543     \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
3544       not permitted\MessageBreak with record=alsoindex package option}%
3545     {You may only use the hybrid \string\makeglossaries[...]\space with
3546       record=off option}%
3547   \else
3548     \edef\@glsxtr@reg@glosslist{#1}%
3549     \ifundef{\glswrite}{\newwrite\glswrite}{}%
3550     \protected@write\@auxout{}{\string\providecommand
3551       \string\@glsorder[1]}{}
3552     \protected@write\@auxout{}{\string\providecommand
3553       \string\@istfilename[1]}{}
3554     \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
3555     \protected@write\@auxout{}{\string\@glsorder{\glsorder}}
3556     \protected@write\@auxout{}{\string\glsxtr@makeglossaries{#1}}
3557     \write\@auxout{\string\providecommand\string\@gls@reference[3]}{}%

```

Iterate through each supplied glossary type and activate it.

```

3558   \@for\@glo@type:=#1\do{%
3559     \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
3560   }%

```

New glossaries must be created before \makeglossaries:

```

3561   \renewcommand*\newglossary[4][]{%
3562     \PackageError{glossaries}{New glossaries
3563       must be created before \string\makeglossaries}{You need
3564       to move \string\makeglossaries\space after all your
3565       \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

3566   \let\@makeglossary\relax
3567   \let\makeglossary\relax
3568   \renewcommand\makeglossaries[1][]{}%

```

Disable all commands that have no effect after \makeglossaries

```

3569   \@disable@onlypremakeg

```

Allow see key:

```

3570   \let\gls@checkseeallowed\relax

```

Adjust \@do@seeglossary. This needs to check for the entries existence.

```

3571   \renewcommand*\@do@seeglossary[2]{%
3572     \glsdoifexists{##1}%
3573   {%
3574     \edef\@gls@label{\glsdetoklabel{##1}}%
3575     \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3576     \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3577     {\@glsxtr@org@doseeglossary{##1}{##2}}%
3578   }%
3579   \@glsxtrwrglossmark
3580   \protected@write\@auxout{}{%

```

```

3581         \string\@gls@reference
3582         {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}%
3583     }%
3584 }%
3585 }%
3586 }%

Adjust \@do@wrglossary
3587 \let\@glsxtr@do@wrglossary\@do@wrglossary
3588 \def\@do@wrglossary{%
3589     \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3590     \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3591     {\@glsxtr@do@wrglossary}%
3592     {\gls@noidxglossary}%
3593 }%

Suppress warning about no \makeglossaries
3594 \let\warn@nomakeglossaries\relax
3595 \def\warn@noprntglossary{%
3596     \GlossariesWarningNoLine{No \string\printglossary\space
3597     or \string\printglossaries\space
3598     found.^^J(Remove \string\makeglossaries\space if you don't want
3599     any glossaries.)^^JThis document will not have a glossary}%
3600 }%

Only warn for glossaries not listed.
3601 \renewcommand{\@gls@noref@warn}[1]{%
3602     \edef\@gls@type{##1}%
3603     \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3604     {%
3605         \GlossariesExtraWarning{Can't use
3606         \string\printnoidxglossary[type={\@gls@type}]
3607         when '\@gls@type' is listed in the optional argument of
3608         \string\makeglossaries}%
3609     }%
3610     {%
3611         \GlossariesWarning{Empty glossary for
3612         \string\printnoidxglossary[type={##1}].
3613         Rerun may be required (or you may have forgotten to use
3614         commands like \string\gls)}}%
3615     }%
3616 }%

Adjust display number list to check for type:
3617 \renewcommand*\@glsdisplaynumberlist[1]{%
3618     \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3619     {\@glsxtr@idx@displaynumberlist{##1}}%
3620     {\@glsxtr@noidx@displaynumberlist{##1}}%
3621 }%

Adjust entry list:

```

```

3622 \renewcommand*\glsentrynumberlist}[1]{%
3623 \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
3624 {\@glxtr@idx@entrynumberlist{##1}}%
3625 {\@glxtr@noidx@entrynumberlist{##1}}%
3626 }%

```

Adjust number list loop

```

3627 \renewcommand*\glsnumberlistloop}[2]{%
3628 \expandafter\DTLifinlist\expandafter{##1}{\@glxtr@reg@glosslist}%
3629 {%
3630 \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3631 not available for glossary ‘##1’}{}%
3632 }%
3633 {\@glxtr@noidx@numberlistloop{##1}{##2}}%
3634 }%

```

Only sanitize sort for normal indexing glossaries.

```

3635 \renewcommand*\glsprestandardsort}[3]{%
3636 \expandafter\DTLifinlist\expandafter{##2}{\@glxtr@reg@glosslist}%
3637 {%
3638 \glsdosanitizesort
3639 }%
3640 {%
3641 \ifglssanitizesort
3642 \@gls@noidx@sanitizesort
3643 \else
3644 \@gls@noidx@nosanitizesort
3645 \fi
3646 }%
3647 }%

```

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

```

3648 \renewcommand*\new@glossaryentry[2]{%
3649 \PackageError{glossaries-extra}{Glossary entries must be defined
3650 in the preamble\MessageBreak when you use the optional argument
3651 of \string\makeglossaries}{Either move your definitions to the
3652 preamble or don’t use the optional argument of
3653 \string\makeglossaries}%
3654 }%

```

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).

```

3655 \let\@glo@assign@sortkey\@glxtr@mixed@assign@sortkey
3656 \renewcommand*\@printgloss@setsort{%

```

Need to extract just the type value.

```

3657 \expandafter\@glxtr@gettype\expandafter,\@glxtr@printglossopts,%
3658 type=\glsdefaulttype,\@end@glxtr@gettype
3659 \def\@glo@sorttype{\@glo@default@sorttype}%
3660 }%

```

Check automake setting:

```

3661 \ifglautomake
3662 \renewcommand*{\@gls@doautomake}{%
3663 \@for\@gls@type:=\@glstr@reg@glosslist\do{%
3664 \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
3665 }%
3666 }%
3667 \fi

```

Check the sort setting (glossaries v4.30 onwards):

```

3668 \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
3669 \fi
3670 }%
3671 \fi
3672 }

```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

`\@printglossary` This no longer simply saves `\@printglossary` with `\let` but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes `\provideignoredglossary` to the `glstex` file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```

3673 \newcommand{\@glstr@orgprintglossary}[2]{%
3674 \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

3675 \def\glossarytitle{%
3676 \ifcsdef{\@glo@type\@glo@type @title}%
3677 {\csuse{\@glo@type\@glo@type @title}}%
3678 {\glossaryname}}%
3679 \def\glossarytoctitle{\glossarytitle}%
3680 \let\org@glossarytitle\glossarytitle
3681 \def\@glossarystyle{%
3682 \ifx\@glossary@default@style\relax
3683 \GlossariesWarning{No default glossary style provided \MessageBreak
3684 for the glossary '\@glo@type'. \MessageBreak
3685 Using deprecated fallback. \MessageBreak
3686 To fix this set the style with \MessageBreak
3687 \string\setglossarystyle\space or use the \MessageBreak
3688 style key=value option}%
3689 \fi
3690 }%
3691 \def\gls@dotoc\toctitle{\glssettoc\toc\toctitle{\@glo@type}}%
3692 \let\@org@glossaryentrynumbers\glossaryentrynumbers
3693 \bgroup
3694 \@printgloss@setsort
3695 \setkeys{printgloss}{#1}%
3696 \ifx\glossarytitle\org@glossarytitle
3697 \else

```

```

3698     \cslet{@glo@type@}@glo@type @title}{\glossarytitle}%
3699     \fi
3700     \let\currentglossary\@glo@type
3701     \let\org@glossaryentrynumbers\glossaryentrynumbers
3702     \let\glsnonextpages\@glsnonextpages
3703     \let\glsnextpages\@glsnextpages

3704     \glsxtractivatenopost
3705     \gls@dotoc@title
3706     \@glossarystyle
3707     \let\gls@org@glossaryentryfield\glossentry
3708     \let\gls@org@glossarysubentryfield\subglossentry
3709     \renewcommand{\glossentry}[1]{%
3710         \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
3711         \gls@org@glossaryentryfield{##1}%
3712     }%
3713     \renewcommand{\subglossentry}[2]{%
3714         \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
3715         \gls@org@glossarysubentryfield{##1}{##2}%
3716     }%
3717     \@gls@preglossaryhook
3718     #2%
3719     \egroup
3720     \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
3721     \global\let\warn@noprintglossary\relax
3722 }

```

xtractivatenopost Change \nopostdesc and \glsxtrnopostpunc to behave as they do in the glossary.

```

3723 \newcommand*{\glsxtractivatenopost}{%
3724     \let\nopostdesc\@nopostdesc
3725     \let\glsxtrnopostpunc\@glsxtr@nopostpunc
3726 }

```

lsxtrnopostpunc

```

3727 \newrobustcmd*{\glsxtrnopostpunc}{%

```

lsxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```

3728 \newcommand{\@glsxtr@nopostpunc}{%
3729     \let\@glsxtr@org@postdescription\glspostdescription
3730     \ifglsnopostdot
3731         \renewcommand{\glspostdescription}{%
3732             \glsnopostdottrue
3733             \let\glspostdescription\@glsxtr@org@postdescription
3734             \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
3735             \glsxtrpostdescription
3736             \@glsxtr@nopostpunc@postdesc}%
3737     \else
3738         \renewcommand{\glspostdescription}{%

```

```

3739 \let\glspostdescription\@glsxtr@org@postdescription
3740 \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
3741 \glsxtrpostdescription
3742 \@glsxtr@nopostpunc@postdesc}%
3743 \fi
3744 \glsnopostdotfalse
3745 }

stpunc@postdesc
3746 \newcommand*{\@glsxtr@nopostpunc@postdesc}{%

restore@postpunc
3747 \newcommand*{\@glsxtr@restore@postpunc}{%
3748 \def\@glsxtr@nopostpunc@postdesc{%
3749 \@glsxtr@org@postdescription
3750 \let\@glsxtr@nopostpunc@postdesc\@empty
3751 \let\glsxtrrestorepostpunc\@empty
3752 }%
3753 }

restorepostpunc Does nothing outside of glossary.
3754 \newcommand*{\glsxtrrestorepostpunc}{%

\@printglossary Redefine.
3755 \renewcommand{\@printglossary}[2]{%
3756 \def\@glsxtr@printglossopts{#1}%
3757 \@glsxtr@orgprintglossary{#1}{#2}%
3758 }

Add a key that switches off the entry targets:
3759 \define@choicekey{printgloss}{target}[\val\nr]{true,false}[true]{%
3760 \ifcase\nr
3761 \let\@glstarget\glsdohypertarget
3762 \else
3763 \let\@glstarget\@secondoftwo
3764 \fi
3765 }

hypernameprefix
3766 \newcommand{\@glsxtrhypernameprefix}{%

New to v1.20:
3767 \define@key{printgloss}{targetnameprefix}{%
3768 \renewcommand{\@glsxtrhypernameprefix}{#1}%
3769 }

glsdohypertarget Redefine to insert \@glsxtrhypernameprefix before the target name.
3770 \let\@glsxtr@org@glsdohypertarget\glsdohypertarget

```

```

3771 \renewcommand{\glsdohypertarget}[2]{%
3772   \@glsxtr@org@glsdohypertarget{\@glsxtrhypernameprefix#1}{#2}%
3773 }

```

@makeglossaries For the benefit of makeglossaries

```

3774 \newcommand*{\glsxtr@makeglossaries}[1]{}

```

@glsxtr@gettype Get just the type.

```

3775 \def\@glsxtr@gettype#1,type=#2,#3\@end@glsxtr@gettype{%
3776   \def\@glo@type{#2}%
3777 }

```

@assign@sortkey Assign the sort key.

```

3778 \newcommand\@glsxtr@mixed@assign@sortkey[1]{%
3779   \edef\@glo@type{\@glo@type}%
3780   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glsxtr@reg@glosslist}%
3781   {%
3782     \@glo@no@assign@sortkey{#1}%
3783   }%
3784   {%
3785     \@glo@assign@sortkey{#1}%
3786   }%
3787 }%

```

Display number list for the regular version:

splaynumberlist

```

3788 \let\@glsxtr@idx@displaynumberlist\glsdisplaynumberlist

```

Display number list for the “noidx” version:

splaynumberlist

```

3789 \newcommand*{\@glsxtr@noidx@displaynumberlist}[1]{%
3790   \letcs{\@gls@loclist}{glo@glsdetoklabel{#1}@loclist}%
3791   \ifdef\@gls@loclist
3792   {%
3793     \def\@gls@noidxloclist@sep{%
3794       \def\@gls@noidxloclist@sep{%
3795         \def\@gls@noidxloclist@sep{%
3796           \glsnumlistsep
3797         }%
3798         \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
3799       }%
3800     }%
3801     \def\@gls@noidxloclist@finalsep{}%
3802     \def\@gls@noidxloclist@prev{}%
3803     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
3804     \@gls@noidxloclist@finalsep
3805     \@gls@noidxloclist@prev
3806   }%
3807   {%

```



```

3808 \glstrundef\tag
3809 \glsdoifexists{#1}%
3810 {%
3811 \GlossariesWarning{Missing location list for ‘#1’. Either
3812 a rerun is required or you haven’t referenced the entry.}%
3813 }%
3814 }%
3815 }%
3816

```

And for the number list loop:

@numberlistloop

```

3817 \newcommand*{\@glstr@noidx@numberlistloop}[3]{%
3818 \letcs{\@gls@loclist}{glo@glstoklabel{#1}@loclist}%
3819 \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
3820 \let\@gls@org@glssseeformat\glssseeformat
3821 \let\glsnoidxdisplayloc#2\relax
3822 \let\glssseeformat#3\relax
3823 \ifdef\@gls@loclist
3824 {%
3825 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
3826 }%
3827 {%
3828 \glstrundef\tag
3829 \glsdoifexists{#1}%
3830 {%
3831 \GlossariesWarning{Missing location list for ‘##1’. Either
3832 a rerun is required or you haven’t referenced the entry.}%
3833 }%
3834 }%
3835 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
3836 \let\glssseeformat\@gls@org@glssseeformat
3837 }%

```

Same for entry number list.

entrynumberlist

```

3838 \newcommand*{\@glstr@noidx@entrynumberlist}[1]{%
3839 \letcs{\@gls@loclist}{glo@glstoklabel{#1}@loclist}%
3840 \ifdef\@gls@loclist
3841 {%
3842 \glsnoidxloclist{\@gls@loclist}%
3843 }%
3844 {%
3845 \glstrundef\tag
3846 \glsdoifexists{#1}%
3847 {%
3848 \GlossariesWarning{Missing location list for ‘#1’. Either

```

```

3849         a rerun is required or you haven't referenced the entry.}%
3850     }%
3851 }%
3852 }%

```

entrynumberlist

```

3853 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

3854 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
3855     \protected@edef\@glsxtr@titlelabel{#1}%
3856     \ifdefvoid\@glsxtr@titlelabel
3857     {%
3858     {%
3859         \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
3860     }%
3861     \ifdefvoid{\@glsxtr@titlelabel}%
3862     {%
3863         \DTLifint{#1}%
3864         {%
3865             \ifnum#1<256\relax
3866                 \edef#2{\char#1\relax}%
3867             \else
3868                 \edef#2{#1}%
3869             \fi
3870         }%
3871         {%
3872             \ifcsundef{#1groupname}%
3873                 {\def#2{#1}}%
3874                 {\letcs#2{#1groupname}}%
3875         }%
3876     }%
3877     {%
3878         \let#2\@glsxtr@titlelabel
3879     }%
3880 }

```

g@getgrouptitle Save original definition of \@gls@getgrouptitle

```

3881 \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.

```

3882 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
3883     \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
3884     \@onelevel@sanitize\@glsxtr@titlelabel
3885     \ifcsdef{\@glsxtr@titlelabel}
3886     {\letcs{#2}{\@glsxtr@titlelabel}}%
3887     {\glsxtr@org@getgrouptitle{#1}{#2}}%
3888 }

```

```

3889 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

trsetgrouptitle  Sets the title for the given group label.
3890 \newcommand{\glsxtrsetgrouptitle}[2]{%
3891   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
3892   \@onelevel@sanitize\@glsxtr@titlelabel
3893   \csxdef{\@glsxtr@titlelabel}{#2}%
3894 }

alsetgrouptitle  As above put only locally defines the title.
3895 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
3896   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
3897   \@onelevel@sanitize\@glsxtr@titlelabel
3898   \csedef{\@glsxtr@titlelabel}{#2}%
3899 }

\glsnavigation  Redefine to use new user-level command.
3900 \renewcommand*{\glsnavigation}{%
3901   \def\@gls@between{%
3902     \ifcsundef{\@gls@hypergroup@list@\@glo@type}%
3903     {%
3904       \def\@gls@list{%
3905       }%
3906       {%
3907         \expandafter\let\expandafter\@gls@list
3908           \csname \@gls@hypergroup@list@\@glo@type\endcsname
3909       }%
3910       \@for\@gls@tmp:=\@gls@list\do{%
3911         \@gls@between
3912         \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
3913         \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
3914         \let\@gls@between\glshypernavsep
3915       }%
3916 }

@noidx@glossary
3917 \renewcommand*{\@print@noidx@glossary}{%
3918   \ifcsdef{\@glsref@\@glo@type}%
3919   {%
3920     \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
3921     {%
3922       \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
3923     }%
3924     {%
3925       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
3926     }%
3927     \glossarysection[\glossarytoctitle]{\glossarytitle}%
3928     \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

3929 \def\@gls@currentlettergroup{%
3930 \begin{theglossary}%
3931 \glossaryheader
3932 \glsresetentrylist
3933 \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}%
3934 \end{theglossary}%
3935 \glossarypostamble
3936 }%
3937 {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

3938 \glsxtrifemptyglossary{\@glo@type}%
3939 }%
3940 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
3941 \@gls@noref@warn{\@glo@type}%
3942 }%
3943 }

```

`noidxdisplayloc` Patch to check for range formations.

```

3944 \renewcommand*{\glsnoidxdisplayloc}[4]{%
3945 \setentrycounter[#1]{#2}%
3946 \@glsxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
3947 }

```

`xtr@display@loc` Patch to check for range formations.

```

3948 \def\@glxtr@display@loc#1#2\end@glxtr@display@loc#3{%
3949 \ifx#1(\relax
3950 \glxtrdisplaystartloc{#2}{#3}%
3951 \else
3952 \ifx#1)\relax
3953 \glxtrdisplayendloc{#2}{#3}%
3954 \else
3955 \glxtrdisplaysingleloc{#1#2}{#3}%
3956 \fi
3957 \fi
3958 }

```

`isplaysingleloc` Single location.

```

3959 \newcommand*{\glxtrdisplaysingleloc}[2]{%
3960 \csuse{#1}{#2}%
3961 }

```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glxtrlocrangefmt`.

displaystartloc Start of a location range.

```
3962 \newcommand*{\glxtrdisplaystartloc}[2]{%
3963   \edef\glxtrlocrangefmt{#1}%
3964   \ifx\glxtrlocrangefmt\empty
3965     \def\glxtrlocrangefmt{\glsnnumberformat}%
3966   \fi
3967   \expandafter\glxtrdisplaysingleloc
3968   \expandafter{\glxtrlocrangefmt}{#2}%
3969 }
```

trdisplayendloc End of a location range.

```
3970 \newcommand*{\glxtrdisplayendloc}[2]{%
3971   \edef\@glxtr@tmp{#1}%
3972   \ifdefempty{\@glxtr@tmp}{\def\@glxtr@tmp{\glsnnumberformat}}{}}%
3973   \ifx\glxtrlocrangefmt\@glxtr@tmp
3974     \else
3975       \GlossariesExtraWarning{Mismatched end location range
3976         (start=\glxtrlocrangefmt, end=\@glxtr@tmp)}%
3977     \fi
3978     \expandafter\glxtrdisplayendlochook\expandafter{\@glxtr@tmp}{#2}%
3979     \expandafter\glxtrdisplaysingleloc
3980     \expandafter{\glxtrlocrangefmt}{#2}%
3981   \def\glxtrlocrangefmt{}%
3982 }
```

splayendlochook Allow the user to hook into the end of range command.

```
3983 \newcommand*{\glxtrdisplayendlochook}[2]{}
```

sxtrlocrangefmt Current range format. Empty if not in a range.

```
3984 \newcommand*{\glxtrlocrangefmt}{}%
```

ls@removespaces Redefine to allow adjustments to location hyperlink.

```
3985 \def\@gls@removespaces#1 #2\@nil{%
3986   \toks@=\expandafter{\the\toks@#1}%
3987   \ifx\@#2\%
3988     \edef\x{\the\toks@}%
3989     \ifx\x\empty
3990       \else
3991         \glxtrlocationhyperlink{\glsenentrycounter}{\@glo@counterprefix}{\the\toks@}%
3992       \fi
3993     \else
3994       \@gls@ReturnAfterFi{%
3995         \@gls@removespaces#2\@nil
3996       }%
3997     \fi
3998 }
```

cationhyperlink

```

3999 \newcommand*{\glxtrlocationhyperlink}[3]{%
4000   \ifdefvoid\glxtrsupplocationurl
4001   {%
4002     \glxtrhyperlink{#1#2#3}{#3}%
4003   }%
4004   {%
4005     \hyperref{\glxtrsupplocationurl}{#1#2#3}{#3}%
4006   }%
4007 }

```

supphypernumber

```

4008 \newcommand*{\glxtrsupphypernumber}[1]{%
4009   {%
4010     \glshasattribute{\glscurrententrylabel}{externallocation}%
4011     {%
4012       \def\glxtrsupplocationurl{%
4013         \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4014     }%
4015     {%
4016       \def\glxtrsupplocationurl{}%
4017     }%
4018     \glshypernumber{#1}%
4019   }%
4020 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

4021 \renewcommand{\@print@glossary}{%
4022   \makeatletter
4023   \@input{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4024   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4025   {%
4026     {\glxtrNoGlossaryWarning{\@glo@type}}%
4027     \ifglxindy
4028       \ifcsundef{@xdy@\@glo@type @language}%
4029       {%
4030         \edef\@do@auxoutstuff{%
4031           \noexpand\AtEndDocument{%
4032             \noexpand\immediate\noexpand\write\@auxout{%
4033               \string\providecommand\string\@xdylanguage[2]{}}%
4034             \noexpand\immediate\noexpand\write\@auxout{%
4035               \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4036           }%
4037         }%
4038       }%
4039     {%
4040       \edef\@do@auxoutstuff{%

```

```

4041      \noexpand\AtEndDocument{%
4042      \noexpand\immediate\noexpand\write\@auxout{%
4043      \string\providecommand\string\@xdylanguage[2]{}}%
4044      \noexpand\immediate\noexpand\write\@auxout{%
4045      \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4046      @language\endcsname}}%
4047      }%
4048      }%
4049      }%
4050      \@do@auxoutstuff
4051      \edef\@do@auxoutstuff{%
4052      \noexpand\AtEndDocument{%
4053      \noexpand\immediate\noexpand\write\@auxout{%
4054      \string\providecommand\string\@gls@codepage[2]{}}%
4055      \noexpand\immediate\noexpand\write\@auxout{%
4056      \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
4057      }%
4058      }%
4059      \@do@auxoutstuff
4060      \fi
4061      \renewcommand*{\@warn@nomakeglossaries}{%
4062      \GlossariesWarningNoLine{\string\makeglossaries\space
4063      hasn't been used,^^Jthe glossaries will not be updated}%
4064      }%
4065      }

```

Setup the warning text to display if the external file for the given glossary is missing.

`\oGlsWarningHead` Header message.

```

4066 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4067   This document is incomplete. The external file associated with
4068   the glossary '#1' (which should be called \texttt{#2})
4069   hasn't been created.%
4070 }

```

`\warningEmptyStart` No entries have been added to the glossary.

```

4071 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4072   This has probably happened because there are no entries defined
4073   in this glossary.%
4074 }

```

`\warningEmptyMain` The default “main” glossary is empty.

```

4075 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4076   If you don't want this glossary,
4077   add \texttt{nomain} to your package option list when you load
4078   \texttt{glossaries-extra.sty}. For example:%
4079 }

```

`\warningEmptyNotMain` A glossary that isn't the default “main” glossary is empty.

```

4080 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4081 Did you forget to use \texttt{type=#1} when you defined your
4082 entries? If you tried to load entries into this glossary with
4083 \texttt{\string\loadglsentries} did you remember to use
4084 \texttt{[#1]} as the optional argument? If you did, check that
4085 the definitions in the file you loaded all had the type set
4086 to \texttt{\string\glsdefaulttype}.%
4087 }

```

WarningCheckFile Advisory message to check the file contents.

```

4088 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4089 Check the contents of the file \texttt{#1}. If
4090 it's empty, that means you haven't indexed any of your entries in this
4091 glossary (using commands like \texttt{\string\gls} or
4092 \texttt{\string\glsadd}) so this list can't be generated.
4093 If the file isn't empty, the document build process hasn't been
4094 completed.%
4095 }

```

WarningAutoMake Message when automake option has been used.

```

4096 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4097 You may need to rerun \LaTeX. If you already have, it may be that
4098 \TeX's shell escape doesn't allow you to run
4099 \ifglxindy xindy\else makeindex\fi. Check the
4100 transcript file \texttt{\jobname.log}. If the shell escape is
4101 disabled, try one of the following:
4102
4103 \begin{itemize}
4104 \item Run the external (Lua) application:
4105
4106 \texttt{makeglossaries-lite.lua \string\jobname\string}
4107
4108 \item Run the external (Perl) application:
4109
4110 \texttt{makeglossaries \string\jobname\string}
4111 \end{itemize}
4112
4113 Then rerun \LaTeX\ on this document.
4114 \GlossariesExtraWarning{Rerun required to build the
4115 glossary '#1' or check \TeX's shell escape allows
4116 you to run \ifglxindy xindy\else makeindex\fi}%
4117 }

```

WarningMisMatch Mismatching \makenoidxglossaries.

```

4118 \newcommand{\GlsXtrNoGlsWarningMisMatch}{%
4119 You need to either replace \texttt{\string\makenoidxglossaries}
4120 with \texttt{\string\makeglossaries} or replace
4121 \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with

```



```

4122 \texttt{\string\printnoidxglossary}
4123 (or \texttt{\string\printnoidxglossaries}) and then rebuild
4124 this document.%
4125 }

```

arningBuildInfo Build advice.

```

4126 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4127   Try one of the following:
4128   \begin{itemize}
4129     \item Add \texttt{automake} to your package option list when you load
4130       \texttt{glossaries-extra.sty}. For example:
4131
4132       \texttt{\string\usepackage[automake]%
4133         \glsopenbrace glossaries-extra\glsclosebrace}
4134
4135     \item Run the external (Lua) application:
4136
4137     \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4138
4139     \item Run the external (Perl) application:
4140
4141     \texttt{makeglossaries \string"\jobname\string"}
4142   \end{itemize}
4143
4144   Then rerun \LaTeX\ on this document.%
4145 }

```

oGlsWarningTail Final paragraph.

```

4146 \newcommand{\GlsXtrNoGlsWarningTail}{%
4147   This message will be removed once the problem has been fixed.%
4148 }

```

GlsWarningNoOut No out file created. Build advice.

```

4149 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4150   The file \texttt{#1} doesn't exist. This most likely means you haven't used
4151   \texttt{\string\makeglossaries} or you have used
4152   \texttt{\string\nofiles}. If this is just a draft version of the
4153   document, you can suppress this message using the
4154   \texttt{\string\nomissingglstext} package option.%
4155 }

```

glossarywarning

```

4156 \newcommand*{@@glxtr@defaultnoglossarywarning}[1]{%
4157   \glossarysection[\glossarytoctitle]{\glossarytitle}
4158   \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@{@@glo@type @in@endcsname}
4159   \par
4160   \glxtrifemptyglossary{#1}%
4161   {%
4162     \GlsXtrNoGlsWarningEmptyStart\space

```

```

4163 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4164 \medskip
4165 \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
4166 \glsopenbrace glossaries-extra\glsclosebrace}
4167 \medskip
4168 }%
4169 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4170 }%
4171 {%
4172 \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4173 {%
4174 \GlsXtrNoGlsWarningCheckFile
4175 {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4176
4177 \ifglsautomake
4178
4179 \GlsXtrNoGlsWarningAutoMake{#1}
4180
4181 \else
4182
4183 \ifthenelse{\equal{#1}{main}}{%
4184 {%
4185 \GlsXtrNoGlsWarningEmptyMain\par
4186 \medskip
4187 \noindent\texttt{\string\usepackage[nomain]%
4188 \glsopenbrace glossaries-extra\glsclosebrace}
4189 \medskip
4190 }%
4191 {}}%
4192
4193 \ifdefequal\makeglossaries\@no@makeglossaries
4194 {%
4195 \GlsXtrNoGlsWarningMisMatch
4196 }%
4197 {%
4198 \GlsXtrNoGlsWarningBuildInfo
4199 }%
4200 \fi
4201 }%
4202 {%
4203 \GlsXtrNoGlsWarningNoOut
4204 {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4205 }%
4206 }%
4207 \par
4208 \GlsXtrNoGlsWarningTail
4209 }

```

Provide some commands to accompany the record option for use with **bib2gls**.

xtresourcefile Since it's dangerous for an external application to create a file with a .tex extension, as from v1.11 this enforces a .glstex extension to avoid conflict.

```
4210 \newcommand*{\glxtrresourcefile}[2] [] {%
```

The record option can't be set after this command.

```
4211 \disable@keys{glossaries-extra.sty}{record}%
4212 \glxtr@writefields
4213 \protected@write\@auxout{\glxtrresourceinit}{\string\glxtr@resource{#1}{#2}}%
4214 \let\@glxtr@org@see@noindex\@gls@see@noindex
4215 \let\@gls@see@noindex\relax
4216 \IfFileExists{#2.glstex}%
4217 {%
```

Can't scope \@input so save and restore the category code of @ to allow for internal commands in the location list.

```
4218 \edef\@bibgls@restreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4219 \makeatletter
4220 \@input{#2.glstex}%
4221 \@bibgls@restreat
4222 }%
4223 {%
4224 \GlossariesExtraWarning{No file '#2.glstex'}%
4225 }%
4226 \let\@gls@see@noindex\@glxtr@org@see@noindex
4227 }
4228 \onlypreamble\glxtrresourcefile
```

xtresourceinit Code used during the protected write operation.

```
4229 \newcommand*{\glxtrresourceinit}{}
```

trresourcecount

```
4230 \newcount\glxtrresourcecount
```

trLoadResources Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.

```
4231 \newcommand*{\GlsXtrLoadResources}[1] [] {%
4232 \ifnum\glxtrresourcecount=0\relax
4233 \glxtrresourcefile[#1]{\jobname}%
4234 \else
4235 \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
4236 \fi
4237 \advance\glxtrresourcecount by 1\relax
4238 }
```

glxtr@resource

```
4239 \newcommand*{\glxtr@resource}[2] {}
```

\glxtr@fields

```
4240 \newcommand*{\glxtr@fields}[1] {}
```

xtr@texencoding

```
4241 \newcommand*{\glxtr@texencoding}[1]{}
```

\glxtr@langtag

```
4242 \newcommand*{\glxtr@langtag}[1]{}
```

@pluralsuffixes

```
4243 \newcommand*{\glxtr@pluralsuffixes}[4]{}
```

tr@shortcutsval

```
4244 \newcommand*{\glxtr@shortcutsval}[1]{}
```

sxtr@linkprefix

```
4245 \newcommand*{\glxtr@linkprefix}[1]{}
```

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```
4246 \newcommand*{\glxtr@writefields}{%
```

```
4247   \protected@write\@auxout{}%
```

```
4248     {\string\providecommand*{\string\glxtr@fields}[1]{}}}%
```

```
4249   \protected@write\@auxout{}%
```

```
4250     {\string\providecommand*{\string\glxtr@resource}[2]{}}}%
```

```
4251   \protected@write\@auxout{}%
```

```
4252     {\string\providecommand*{\string\glxtr@pluralsuffixes}[4]{}}}%
```

```
4253   \protected@write\@auxout{}%
```

```
4254     {\string\providecommand*{\string\glxtr@shortcutsval}[1]{}}}%
```

```
4255   \protected@write\@auxout{}%
```

```
4256     {\string\providecommand*{\string\glxtr@linkprefix}[1]{}}}%
```

```
4257   \protected@write\@auxout{}{\string\glxtr@fields{\@gls@keymap}}}%
```

```
4258   \protected@write\@auxout{}%
```

```
4259     {\string\providecommand*{\string\glxtr@record}[5]{}}}%
```

If any languages have been loaded, the language tag will be available in `\CurrentTrackedLanguageTag` (provided by `tracklang`). For multilingual documents, the required locale will have to be indicated in the sort key when using `\glxtrresourcefile`.

```
4260 \ifdef\CurrentTrackedLanguageTag
```

```
4261 {%
```

```
4262   \protected@write\@auxout{}%
```

```
4263     {\string\glxtr@langtag{\CurrentTrackedLanguageTag}}%
```

```
4264   }%
```

```
4265   {%
```

```
4266     \protected@write\@auxout{}{\string\glxtr@pluralsuffixes
```

```
4267       {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
```

```
4268       {\glxtrabbrvpluralsuffix}}%
```

```
4269   \ifdef\inputencodingname
```

```
4270   {%
```

```
4271     \protected@write\@auxout{}{\string\glxtr@texencoding{\inputencodingname}}%
```

```
4272   }%
```

```
4273   {%
```

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.)

```
4274 \ifpackageloaded{fontspec}%
4275 {\protected@write\@auxout{}\string\glsxtr@texencoding{utf8}}}%
4276 {}%
4277 }%
4278 \protected@write\@auxout{}\string\glsxtr@shortcutsval{\@glsxtr@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.

```
4279 \AtBeginDocument
4280 {\protected@write\@auxout{}\string\glsxtr@linkprefix{\glolinkprefix}}}%
4281 \let\glsxtr@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.

```
4282 \ifglsautomake
4283 \IfFileExists{\jobname.aux}%
4284 {\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`.

```
4285 \ifx\@gls@doautomake\@gls@doautomake@err
4286 \let\@gls@doautomake\relax
4287 \fi
4288 \fi
4289 }
```

`do@automake@err`

```
4290 \newcommand*{\@gls@doautomake@err}{%
4291 \PackageError{glossaries}{You must use
4292 \string\makeglossaries\space with automake=true}
4293 {%
4294 Either remove the automake=true setting or
4295 add \string\makeglossaries\space to your document preamble.%
4296 }%
4297 }
```

Allow locations specific to a particular counter to be recorded.

`\glsxtr@record`

```
4298 \newcommand*{\glsxtr@record}[5]{}
```

`r@counterrecord` Aux file command.

```
4299 \newcommand*{\glsxtr@counterrecord}[3]{%
4300 \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
4301 }
```

unterrecordhook Hook used by \@glsxtr@dorecord.

```
4302 \newcommand*{\@glsxtr@counterrecordhook}{}
```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```
4303 \newcommand*{\GlsXtrRecordCounter}[1]{%
```

```
4304   \@glsxtr@recordcounter{#1}}%
```

```
4305 }
```

```
4306 \@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
4307 \newcommand*{\@glsxtr@docounterrecord}[1]{%
```

```
4308   \protected@write\@auxout{}{\string\glsxtr@counterrecord
```

```
4309     {\@gls@label}{#1}{\csuse{the#1}}}%
```

```
4310 }
```

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.

```
4311 \newcommand*{\glsxtrglossentry}[1]{%
```

```
4312   \glsxtrtitleorpdforheading
```

```
4313   {\@glsxtrglossentry{#1}}%
```

```
4314   {\glsentryname{#1}}%
```

```
4315   {\glsxtrheadname{#1}}%
```

```
4316 }
```

lsxtrglossentry Another test is needed in case \@glsxtrglossentry has been written to the table of contents.

```
4317 \newrobustcmd*{\@glsxtrglossentry}[1]{%
```

```
4318   \glsxtrtitleorpdforheading
```

```
4319   {%
```

```
4320     \glsdoifexists{#1}}%
```

```
4321     {%
```

```
4322       \begingroup
```

```
4323         \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
```

```
4324         \edef\currentglossary{\glsentrytype{\glscurrententrylabel}}%
```

```
4325         \ifglshasparent{#1}%
```

```
4326         {\glssubentryitem{#1}}%
```

```
4327         {\glsentryitem{#1}}%
```

```
4328         \glstarget{#1}{\glssentryname{#1}}%
```

```
4329       \endgroup
```

```
4330     }%
```

```
4331   }%
```

```
4332   {\glssentryname{#1}}%
```

```
4333   {\glsxtrheadname{#1}}%
```

```
4334 }
```

`glossentryother` As `\glxtrglossentry` but uses a different field. First argument is command 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.

```

4335 \newcommand*{\glxtrglossentryother}[3]{%
4336   \ifstrempy{#1}%
4337   {%
4338     \ifcsdef{glxtrhead#3}%
4339     {%
4340       \glxtrtitleorpdforheading
4341       {\@glxtrglossentryother{#2}{#3}{#1}}%
4342       {\@gls@entry@field{#2}{#3}}%
4343       {\csuse{glxtrhead#3}{#2}}%
4344     }%
4345     {%
4346       \glxtrtitleorpdforheading
4347       {\@glxtrglossentryother{#2}{#3}{#1}}%
4348       {\@gls@entry@field{#2}{#3}}%
4349       {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
4350     }%
4351   }%
4352   {%
4353     \glxtrtitleorpdforheading
4354     {\@glxtrglossentryother{#2}{#3}{#1}}%
4355     {\@gls@entry@field{#2}{#3}}%
4356     {#1}%
4357   }%
4358 }
```

`glossentryother` As `\@glxtrglossentry` but uses a different field.

```

4359 \newrobustcmd*{\@glxtrglossentryother}[3]{%
4360   \glxtrtitleorpdforheading
4361   {%
4362     \glsdoidexists{#1}%
4363     {%
4364       \begingroup
4365       \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4366       \edef\currentglossary{\glsenentrytype{\glscurrententrylabel}}%
4367       \ifglshasparent{#1}%
4368       {\glssubentryitem{#1}}%
4369       {\glsenentryitem{#1}}%
4370       \glstarget{#1}{\glossentrynameother{#1}{#2}}%
4371     \endgroup
4372   }%
4373   }%
4374   {\@gls@entry@field{#1}{#2}}%
4375   {#3}%
4376 }
```

`\printunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```
4377 \newcommand*{\printunsrtglossary}{%
4378   \@ifstar\s@printunsrtglossary\@printunsrtglossary
4379 }
```

`\printunsrtglossary` Unstarred version.

```
4380 \newcommand*{\@printunsrtglossary}[1][]{%
4381   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4382 }
```

`\printunsrtglossary` Starred version.

```
4383 \newcommand*{\s@printunsrtglossary}[2][]{%
4384   \begingroup
4385     #2%
4386     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4387   \endgroup
4388 }
```

`\printunsrtglossaries` Similar to `\printnoidxglossaries` but it displays all entries defined for the given glossary without sorting.

```
4389 \newcommand*{\printunsrtglossaries}{%
4390   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
4391 }
```

`\@print@unsrt@glossary`

```
4392 \newcommand*{\@print@unsrt@glossary}{%
4393   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4394   \glossarypreamble
4395   check for empty list
4396   \glsxtrifemptyglossary{\@glo@type}%
4397   {%
4398     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
4399   }%
4400   \key@ifundefined{glossentry}{group}%
4401   {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
4402   {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
4403   \def\@gls@currentlettergroup{}}%
```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```
4404 \def\@glsxtr@doglossary{%
4405   \begin{theglossary}%
4406   \glossaryheader
4407   \glsresetentrylist
4408 }%
4409 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
```



```

4410      :\expandafter=\csname glolist@\@glo@type\endcsname\do{%
4411      \ifdefempty{\glscurrententrylabel}
4412      {}%
4413      {%

```

Provide a hook (for example to measure width).

```

4414      \let\glxtr@process\@firstofone
4415      \let\printunsrtglossaryskipentry
4416      \@glxtr@printunsrtglossaryskipentry
4417      \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%

```

Don't check group for child entries.

```

4418      \glxtr@process
4419      {%
4420      \ifglshasparent{\glscurrententrylabel}{}%
4421      {%
4422      \@glxtr@checkgroup\glscurrententrylabel
4423      \expandafter\appto\expandafter\@glxtr@doglossary\expandafter
4424      {\@glxtr@groupheading}%
4425      }%
4426      \eappto\@glxtr@doglossary{%
4427      \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
4428      }%
4429      }%
4430      }%
4431      \appto\@glxtr@doglossary{\end{theglossary}}%
4432      \printunsrtglossarypredoglossary
4433      \@glxtr@doglossary
4434      }%
4435      \glossarypostamble
4436      }

```

entryprocesshook

```

4437 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{}

```

entryprocesshook

```

4438 \newcommand*{\printunsrtglossaryskipentry}{%
4439   \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
4440   can only be used within \string\printunsrtglossaryentryprocesshook}{}%
4441 }

```

entryprocesshook

```

4442 \newcommand*{\@glxtr@printunsrtglossaryskipentry}{%
4443   \let\glxtr@process\@gobble
4444 }

```

rypredoglossary

```

4445 \newcommand*{\printunsrtglossarypredoglossary}{}

```

lossary@handler

```
4446 \newcommand{\@printunsrt@glossary@handler}[1]{%
4447   \xdef\glscurrententrylabel{#1}%
4448   \printunsrtglossaryhandler\glscurrententrylabel
4449 }
```

glossaryhandler

```
4450 \newcommand{\printunsrtglossaryhandler}[1]{%
4451   \glxtrunsrtdo{#1}%
4452 }
```

triflabelinlist Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of `\@gls@ifinlist` which ensures the label and list are fully expanded.

```
4453 \newrobustcmd*{\glxtriflabelinlist}[4]{%
4454   \protected@edef\glxtr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
4455   \@glxtr@doiflabelinlist{#3}{#4}%
4456 }
```

srtglossaryunit

```
4457 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
4458   \s@printunsrtglossary[type=\gldefaulttype,#1]{%
4459     \printunsrtglossaryunitsetup{#2}%
4460   }%
4461 }
```

ossaryunitsetup

```
4462 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
4463   \renewcommand{\printunsrtglossaryhandler}[1]{%
4464     \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
4465     {\glxtrunsrtdo{##1}}%
4466     {}%
4467   }%
```

Only the target names should have the prefixes adjusted as `\gls` etc need the original `\glolinkprefix`. The `\@gobble` part discards `\glolinkprefix`.

```
4468   \ifcsundef{theH#1}%
4469   {%
4470     \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
4471   }%
4472   {%
4473     \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
4474   }%
4475   \renewcommand*{\glossarysection}[2][ ]{%
4476     \appto\glossarypostamble{\glspare\medskip\glspare}%
4477 }
```

srtglossaryunit

```

4478 \newcommand{\print@noop@unsrtglossaryunit}[2][]{%
4479   \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
4480     requires the record=only or record=alsoindex package option}{}%
4481 }

```

t@getgrouptitle

```

4482 \newrobustcmd*{\@glxtr@unsrt@getgrouptitle}[2]{%
4483   \protected@edef\@glxtr@titlelabel{glxtr@grouptitle@#1}%
4484   \@onelevel@sanitize\@glxtr@titlelabel
4485   \ifcsdef{\@glxtr@titlelabel}
4486     {\letcs{#2}{\@glxtr@titlelabel}}%
4487     {\def#2{#1}}%
4488 }

```

\glxtrunsrtdo Provide a user-level call to \@glxtr@noidx@do to make it easier to define a new handler.

```

4489 \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).

```

4490 \newcommand*{\glxtrgroupfield}{group}

```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while \@glxtr@doglossary is being constructed rather than in the handler.

lsxtr@checkgroup The argument is the entry's label. (This block of code was formerly in \@glxtr@noidx@do.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in \@glxtr@groupheading, which will be empty if no heading is required.

```

4491 \newcommand*{\@glxtr@checkgroup}[1]{%
4492   \def\@glxtr@groupheading{}%
4493   \key@ifundefined{glossentry}{group}%
4494   {%
4495     \letcs{\@gls@sort}{glo@\glstdetoklabel{#1}@sort}%
4496     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
4497   }%
4498   {%
4499     \protected@edef\@glo@thislettergrp{%
4500       \csuse{glo@\glstdetoklabel{#1}@\glxtrgroupfield}}%
4501   }%
4502   \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
4503   {}%
4504   {%
4505     \ifdefempty{\@gls@currentlettergroup}{}%
4506     {\def\@glxtr@groupheading{\glsgroupskip}}%
4507     \eappto\@glxtr@groupheading{%

```

```

4508     \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
4509 }%
4510 }%
4511 \let\@gls@currentlettergroup\@glo@thislettergrp
4512 }

```

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.

```

4513 \newcommand{\@glsxtr@noidx@do}[1]{%
4514   \ifglsentryexists{#1}%
4515   {%
4516     \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4517     \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@location}%
4518     \ifglshasparent{#1}%
4519     {%
4520       \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
4521       \ifdefvoid{\@gls@location}%
4522       {%
4523         \ifdefvoid{\@gls@loclist}%
4524         {%
4525           \subglossentry{\gls@level}{#1}{}%
4526         }%
4527         {%
4528           \subglossentry{\gls@level}{#1}%
4529           {%
4530             \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4531           }%
4532         }%
4533       }%
4534       {%
4535         \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
4536       }%
4537     }%
4538   {%
4539     \ifdefvoid{\@gls@location}%
4540     {%
4541       \ifdefvoid{\@gls@loclist}
4542       {%
4543         \glossentry{#1}{}%
4544       }%
4545       {%
4546         \glossentry{#1}%
4547         {%
4548           \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4549         }%
4550       }%
4551     }%
4552     {%
4553       \glossentry{#1}%

```

```

4554      {%
4555      \glossaryentrynumbers{\@gls@location}%
4556      }%
4557    }%
4558  }%
4559 }%
4560 {}%
4561 }

```

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

```
4562 \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>}
```

```

4563 \newcommand*{\@glsxtrnewgls}[4]{%
4564   \ifdef{#3}%
4565   {%
4566     \PackageError{glossaries-extra}{Command \string#3\space already
4567 defined}{}%
4568   }%
4569   {%
4570     \ifcsdef{@#4like@#2}%
4571     {%
4572       \advance\@glsxtrnewgls@inner by \@ne
4573       \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
4574     }%
4575     {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
4576     \expandafter\newrobustcmd\expandafter*\expandafter
4577     #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
4578     \ifstrempy{#1}%
4579     {%
4580       \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][ ]{%
4581         \new@ifnextchar[%
4582         {\csname @#4@\endcsname{##1}{#2##2}}%
4583         {\csname @#4@\endcsname{##1}{#2##2} [ ]}%
4584       }%
4585     }%
4586   }%

```

```

4587 \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2] [] {%
4588 \new@ifnextchar [%
4589 {\csname @#4@\endcsname{#1,##1}{#2##2}}%
4590 {\csname @#4@\endcsname{#1,##1}{#2##2} []}%
4591 }%
4592 }%
4593 }%
4594 }

```

`\glsxtrnewgls` `\glsxtrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```

4595 \newrobustcmd*{\glsxtrnewgls}[3] [] {%
4596 \@glsxtrnewgls{#1}{#2}{#3}{gls}%
4597 }

```

`\glsxtrnewglslike` Provide a way to conveniently define commands that behave like `\gls`, `\glspl`, `\Gls` and `\Glspl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

4598 \newrobustcmd*{\glsxtrnewglslike}[6] [] {%
4599 \@glsxtrnewgls{#1}{#2}{#3}{gls}%
4600 \@glsxtrnewgls{#1}{#2}{#4}{glspl}%
4601 \@glsxtrnewgls{#1}{#2}{#5}{Gls}%
4602 \@glsxtrnewgls{#1}{#2}{#6}{Glspl}%
4603 }

```

`\glsxtrnewGLSlike` Provide a way to conveniently define commands that behave like `\GLS`, `\GLSpl` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

4604 \newrobustcmd*{\glsxtrnewGLSlike}[4] [] {%
4605 \@glsxtrnewgls{#1}{#2}{#3}{GLS}%
4606 \@glsxtrnewgls{#1}{#2}{#4}{GLSpl}%
4607 }

```

`\glsxtrnewrgls` As `\glsxtrnewgls` but for `\rgls`.

```

4608 \newrobustcmd*{\glsxtrnewrgls}[3] [] {%
4609 \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
4610 }

```

`\glsxtrnewrglslike` As `\glsxtrnewglslike` but for `\rgls` etc.

```

4611 \newrobustcmd*{\glsxtrnewrglslike}[6] [] {%
4612 \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
4613 \@glsxtrnewgls{#1}{#2}{#4}{rglspl}%
4614 \@glsxtrnewgls{#1}{#2}{#5}{rGls}%
4615 \@glsxtrnewgls{#1}{#2}{#6}{rGlspl}%
4616 }

```

`\sxttrnewrGLSlike` As `\glxtrnewGLSlike` but for `\rGLS` etc.

```
4617 \newrobustcmd*{\glxtrnewrGLSlike}[4][\]{%
4618   \@glxtrnewgls{#1}{#2}{#3}{rGLS}%
4619   \@glxtrnewgls{#1}{#2}{#4}{rGLSpl}%
4620 }
```

Provide easy access to record count fields.

`\totalRecordCount` Access total record count. This is designed to be expandable. The argument is the label.

```
4621 \newcommand*{\GlsXtrTotalRecordCount}[1]{%
4622   \ifcsdef{glo@\glsetoklabel{#1}@recordcount}%
4623   {\csname glo@\glsetoklabel{#1}@recordcount\endcsname}%
4624   {0}%
4625 }
```

`\sXtrRecordCount` Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
4626 \newcommand*{\GlsXtrRecordCount}[2]{%
4627   \ifcsdef{glo@\glsetoklabel{#1}@recordcount.#2}%
4628   {\csname glo@\glsetoklabel{#1}@recordcount.#2\endcsname}%
4629   {0}%
4630 }
```

`\tionRecordCount` Access record count for a particular counter and location. The first argument is the label. The second argument is the counter name. The third argument is the location. This command shouldn't be used if the location doesn't fully expand unless `\glxtrdetoklocation` can be set to something sensible.

```
4631 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
4632   \ifcsdef{glo@\glsetoklabel{#1}@recordcount.#2.\glxtrdetoklocation{#3}}%
4633   {\csname glo@\glsetoklabel{#1}@recordcount.#2.\glxtrdetoklocation{#3}\endcsname}%
4634   {0}%
4635 }
```

`\trdetoklocation`

```
4636 \newcommand*{\glxtrdetoklocation}[1]{#1}
```

`\ablerecordcount`

```
4637 \newcommand*{\glxtrenablerecordcount}{%
4638   \renewcommand*{\gls}{\rgls}%
4639   \renewcommand*{\Gls}{\rGls}%
4640   \renewcommand*{\glspl}{\rglspl}%
4641   \renewcommand*{\Glspl}{\rGlspl}%
4642   \renewcommand*{\GLS}{\rGLS}%
4643   \renewcommand*{\GLSpl}{\rGLSpl}%
4644 }
```

`\ordtriggervalue` The value used by the record trigger test. The argument is the entry's label.

```
4645 \newcommand*{\glxtrrecordtriggervalue}[1]{%
```

```

4646 \GlsXtrTotalRecordCount{#1}%
4647 }

```

dCountAttribute

```

4648 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
4649 \@for\@glxstr@cat:=#1\do
4650 {%
4651 \ifdefempty{\@glxstr@cat}{}%
4652 {%
4653 \glsssetcategoryattribute{\@glxstr@cat}{recordcount}{#2}%
4654 }%
4655 }%
4656 }

```

trifrecordtrigger

```
\glxstrifrecordtrigger{<label>}{<trigger format>}{<normal>}
```

```

4657 \newcommand*{\glxstrifrecordtrigger}[3]{%
4658 \glshasattribute{#1}{recordcount}%
4659 {%
4660 \ifnum\glxstrrecordtriggervalue{#1}>\glsggetattribute{#1}{recordcount}\relax
4661 #3%
4662 \else
4663 #2%
4664 \fi
4665 }%
4666 {#3}%
4667 }

```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```

4668 \newcommand*{\@glxstr@rglstrigger@record}[3]{%
4669 \edef\glslabel{\glsdetoklabel{#2}}%
4670 \let\@gls@link@label\glslabel
4671 \def\@glxstr@thevalue{%
4672 \def\@glxstr@theHvalue{\@glxstr@thevalue}%
4673 \def\@glsnumberformat{glstriggerrecordformat}%
4674 \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
4675 \edef\glstype{\csname glo@\glslabel @type\endcsname}%
4676 \def\@glxstr@thevalue{%
4677 \def\@glxstr@theHvalue{\@glxstr@thevalue}%
4678 \glxstrinitwrgloss
4679 \glslinkpresetkeys
4680 \setkeys{glslink}{#1}%
4681 \glslinkpostsetkeys
4682 \ifdefempty{\@glxstr@thevalue}%
4683 {%
4684 \@gls@saveentrycounter

```



```

4685 }%
4686 {%
4687   \let\theglsentrycounter\@glstr@thevalue
4688   \def\theHglentrycounter{\@glstr@theHvalue}%
4689 }%
4690 \ifglstrinitwrglossbefore
4691   \@do@wrglossary{#2}%
4692 \fi
4693 #3%
4694 \ifglstrinitwrglossbefore
4695 \else
4696   \@do@wrglossary{#2}%
4697 \fi
4698 \ifKV@glslink@local
4699   \glsllocalunset{#2}%
4700 \else
4701   \glsunset{#2}%
4702 \fi
4703 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

4704 \newcommand*{\glstriggerrecordformat}[1]{%

```

\rgls

```

4705 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

\@rgls

```

4706 \newcommand*{\@rgls}[2][{}]{%
4707   \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
4708 }

```

\@rgls@

```

4709 \def\@rgls@#1#2[#3]{%
4710   \glstrifrecordtrigger{#2}%
4711   {%
4712     \@glstr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
4713   }%
4714   {%
4715     \@gls@{#1}{#2}[#3]%
4716   }%
4717 }%

```

\rglsp1

```

4718 \newrobustcmd*{\rglsp1}{\@gls@hyp@opt\@rglsp1}

```

\@rglsp1

```

4719 \newcommand*{\@rglsp1}[2][{}]{%

```

```

4720 \new@ifnextchar[{\@rglsp1@{#1}{#2}}{\@rglsp1@{#1}{#2}[]}%
4721 }

```

\@rglsp1@

```

4722 \def\@rglsp1@#1#2[#3]{%
4723   \glxtrifrecordtrigger{#2}%
4724   {%
4725     \@glxtr@rglstrigger@record{#1}{#2}{\rglsp1format{#2}{#3}}%
4726   }%
4727   {%
4728     \@rglsp1@{#1}{#2}[#3]%
4729   }%
4730 }%

```

\rGls

```

4731 \newrobustcmd*{\rGls}{\@gls@hyp@opt\rGls}

```

\@rGls

```

4732 \newcommand*{\@rGls}[2][]{%
4733   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
4734 }

```

\@rGls@

```

4735 \def\@rGls@#1#2[#3]{%
4736   \glxtrifrecordtrigger{#2}%
4737   {%
4738     \@glxtr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
4739   }%
4740   {%
4741     \@Gls@{#1}{#2}[#3]%
4742   }%
4743 }%

```

\rGlspl

```

4744 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\rGlspl}

```

\@rGlspl

```

4745 \newcommand*{\@rGlspl}[2][]{%
4746   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
4747 }

```

\@rGlspl@

```

4748 \def\@rGlspl@#1#2[#3]{%
4749   \glxtrifrecordtrigger{#2}%
4750   {%
4751     \@glxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
4752   }%
4753   {%

```

```

4754 \@GLspl@{#1}-{#2}[#3]%
4755 }%
4756 }%

```

\rGLS

```

4757 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\rGLS}

```

\@rGLS

```

4758 \newcommand*{\@rGLS}[2][{}]{%
4759 \new@ifnextchar[{\@rGLS@{#1}-{#2}}{\@rGLS@{#1}-{#2}[]}%
4760 }

```

\@rGLS@

```

4761 \def\rGLS@#1#2[#3]{%
4762 \glsxtrifrecordtrigger{#2}%
4763 {%
4764 \@glsxtr@rglstrigger@record{#1}-{#2}{\rGLSformat{#2}{#3}}%
4765 }%
4766 {%
4767 \@GLS@{#1}-{#2}[#3]%
4768 }%
4769 }%

```

\rGLSpl

```

4770 \newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\rGLSpl}

```

\@rGLSpl

```

4771 \newcommand*{\@rGLSpl}[2][{}]{%
4772 \new@ifnextchar[{\@rGLSpl@{#1}-{#2}}{\@rGLSpl@{#1}-{#2}[]}%
4773 }

```

\@rGLSpl@

```

4774 \def\rGLSpl@#1#2[#3]{%
4775 \glsxtrifrecordtrigger{#2}%
4776 {%
4777 \@glsxtr@rglstrigger@record{#1}-{#2}{\rGLSplformat{#2}{#3}}%
4778 }%
4779 {%
4780 \@GLSpl@{#1}-{#2}[#3]%
4781 }%
4782 }%

```

\rglsformat

```

4783 \newcommand*{\rglsformat}[2]{%
4784 \glsifregular{#1}
4785 {\glsentryfirst{#1}}%
4786 {\ifglsashaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
4787 }

```

```

\rglsplformat
4788 \newcommand*{\rglsplformat}[2]{%
4789   \glsifregular{#1}
4790   {\glsentryfirstplural{#1}}%
4791   {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
4792 }

\rGlsformat
4793 \newcommand*{\rGlsformat}[2]{%
4794   \glsifregular{#1}
4795   {\Glsentryfirst{#1}}%
4796   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
4797 }

\rGlsplformat
4798 \newcommand*{\rGlsplformat}[2]{%
4799   \glsifregular{#1}
4800   {\Glsentryfirstplural{#1}}%
4801   {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
4802 }

\rGLSformat
4803 \newcommand*{\rGLSformat}[2]{%
4804   \expandafter\mfirstucMakeUppercase\expandafter{\rglsformat{#1}{#2}}%
4805 }

\rGLSplformat
4806 \newcommand*{\rGLSplformat}[2]{%
4807   \expandafter\mfirstucMakeUppercase\expandafter{\rglsplformat{#1}{#2}}%
4808 }

```

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@glxtr@linkcount@<label>` where `<label>` is the entry’s label. Since this is performed within `\@gls@link` the label can be accessed with `\glslabel`.

```

4809 \newcommand{\@glsxtr@do@inc@linkcount}{%
  Does this entry have the linkcount attribute set?
4810   \glsifattribute{\glslabel}{linkcount}{true}%
4811   {%

```

Does the counter exist?

```
4812 \ifcsdef{c@glxtr@linkcount@glslabel}{}%  
4813 {%
```

Counter doesn't exist, so define it.

```
4814 \newcounter{glxtr@linkcount@glslabel}%
```

If linkcountmaster is set, add to counter reset.

```
4815 \glshasattribute{glslabel}{linkcountmaster}%  
4816 {%
```

Need to ensure values are fully expanded.

```
4817 \begingroup  
4818 \edef\x{\endgroup\noexpand\@addtoreset{glxtr@linkcount@glslabel}%  
4819 {\glsggetattribute{glslabel}{linkcountmaster}}}%  
4820 \x  
4821 }%  
4822 {}%  
4823 }%
```

Increment counter:

```
4824 \glxtrinlinkcounter{glxtr@linkcount@glslabel}%  
4825 }%  
4826 {}%  
4827 }
```

`\rincLinkCounter` May be redefined to use `\refstepcounter` if required.

```
4828 \newcommand*{\glxtrinlinkcounter}[1]{\stepcounter{#1}}
```

`\linkCounterValue` Expands to the associated link counter register or 0 if not defined.

```
4829 \newcommand*{\GlsXtrLinkCounterValue}[1]{%  
4830 \ifcsundef{c@glxtr@linkcount@#1}{0}{\csname c@glxtr@linkcount@#1\endcsname}%  
4831 }
```

`\rTheLinkCounter` Expands to the display value of the associated link counter or 0 if not defined.

```
4832 \newcommand*{\GlsXtrTheLinkCounter}[1]{%  
4833 \ifcsundef{theglsxtr@linkcount@#1}{0}%  
4834 {\csname theglxtr@linkcount@#1\endcsname}%  
4835 }
```

`\ifLinkCounterDef` Tests if the counter has been defined

```
4836 \newcommand*{\GlsXtrIfLinkCounterDef}[3]{%  
4837 \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%  
4838 }
```

`\LinkCounterName` Expands to the associated link counter name. (No check for existence.)

```
4839 \newcommand*{\GlsXtrLinkCounterName}[1]{glxtr@linkcount@#1}
```

ableLinkCounting `\GlsXtrEnableLinkCounting[(master counter)]{(categories)}`

Enable link counting for the given categories.

```

4840 \newcommand*{\GlsXtrEnableLinkCounting}[2][]{%
4841   \let\glstr@inc@linkcount\@glstr@do@inc@linkcount
4842   \@for\@glstr@label:=#2\do
4843   {%
4844     \glsssetcategoryattribute{\@glstr@label}{linkcount}{true}%
4845     \ifstrempy{#1}{}%
4846     {%
4847       \ifcsundef{c@#1}%
4848       {\@nocounterr{#1}}%
4849       {\glsssetcategoryattribute{\@glstr@label}{linkcountmaster}{#1}}%
4850     }%
4851   }%
4852 }
4853 \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.

```

4854 \ifpackageloaded{glossaries-accsupp}
4855 {

```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```

4856   \newcommand*{\glsaccessname}[1]{%
4857     \glsnameaccessdisplay
4858     {%
4859       \glssentryname{#1}%
4860     }%
4861     {#1}%
4862   }

```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

4863   \newcommand*{\Glsaccessname}[1]{%
4864     \glsnameaccessdisplay
4865     {%
4866       \Glsentryname{#1}%
4867     }%
4868     {#1}%
4869   }

```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```
4870 \newcommand*{\GLSaccessname}[1]{%
4871   \glanameaccessdisplay
4872   {%
4873     \mfirstucMakeUppercase{\glentryname{#1}}%
4874   }%
4875   {#1}%
4876 }
```

`\glsaccesstext` Display the text value (no link and no check for existence).

```
4877 \newcommand*{\glsaccesstext}[1]{%
4878   \glstextaccessdisplay
4879   {%
4880     \glentrytext{#1}%
4881   }%
4882   {#1}%
4883 }
```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```
4884 \newcommand*{\Glsaccesstext}[1]{%
4885   \glstextaccessdisplay
4886   {%
4887     \Glsentrytext{#1}%
4888   }%
4889   {#1}%
4890 }
```

`\GLSAccessstext` Display the text value (no link and no check for existence) converted to upper case.

```
4891 \newcommand*{\GLSAccessstext}[1]{%
4892   \glstextaccessdisplay
4893   {%
4894     \mfirstucMakeUppercase{\glentrytext{#1}}%
4895   }%
4896   {#1}%
4897 }
```

`glsaccessplural` Display the plural value (no link and no check for existence).

```
4898 \newcommand*{\glsaccessplural}[1]{%
4899   \glspluralaccessdisplay
4900   {%
4901     \glentryplural{#1}%
4902   }%
4903   {#1}%
4904 }
```

`Glsaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

4905 \newcommand*{\Glsaccessplural}[1]{%
4906   \glspluralaccessdisplay
4907   {%
4908     \Glsentryplural{#1}%
4909   }%
4910   {#1}%
4911 }

```

\Glsaccessplural Display the plural value (no link and no check for existence) converted to upper case.

```

4912 \newcommand*{\Glsaccessplural}[1]{%
4913   \glspluralaccessdisplay
4914   {%
4915     \mfirstucMakeUppercase{\glsentryplural{#1}}%
4916   }%
4917   {#1}%
4918 }

```

\glsaccessfirst Display the first value (no link and no check for existence).

```

4919 \newcommand*{\glsaccessfirst}[1]{%
4920   \glsfirstaccessdisplay
4921   {%
4922     \glsentryfirst{#1}%
4923   }%
4924   {#1}%
4925 }

```

\Glsaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

4926 \newcommand*{\Glsaccessfirst}[1]{%
4927   \glsfirstaccessdisplay
4928   {%
4929     \Glsentryfirst{#1}%
4930   }%
4931   {#1}%
4932 }

```

\Glsaccessfirst Display the first value (no link and no check for existence) converted to upper case.

```

4933 \newcommand*{\Glsaccessfirst}[1]{%
4934   \glsfirstaccessdisplay
4935   {%
4936     \mfirstucMakeUppercase{\glsentryfirst{#1}}%
4937   }%
4938   {#1}%
4939 }

```

\glsaccessfirstplural Display the firstplural value (no link and no check for existence).

```

4940 \newcommand*{\glsaccessfirstplural}[1]{%
4941   \glsfirstpluralaccessdisplay

```



```

4942   {%
4943       \glstentryfirstplural{#1}%
4944   }%
4945   {#1}%
4946 }

```

glsaccessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

4947 \newcommand*{\Glsaccessfirstplural}[1]{%
4948     \glstfirstpluralaccessdisplay
4949     {%
4950         \glstentryfirstplural{#1}%
4951     }%
4952     {#1}%
4953 }

```

GLSaccessfirstplural Display the firstplural value (no link and no check for existence) converted to upper case.

```

4954 \newcommand*{\GLSaccessfirstplural}[1]{%
4955     \glstfirstpluralaccessdisplay
4956     {%
4957         \mfirstucMakeUppercase{\glstentryfirstplural{#1}}%
4958     }%
4959     {#1}%
4960 }

```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```

4961 \newcommand*{\glsaccesssymbol}[1]{%
4962     \glssymbolaccessdisplay
4963     {%
4964         \glstentrysymbol{#1}%
4965     }%
4966     {#1}%
4967 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

4968 \newcommand*{\GLSaccesssymbol}[1]{%
4969     \glssymbolaccessdisplay
4970     {%
4971         \glstentrysymbol{#1}%
4972     }%
4973     {#1}%
4974 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

4975 \newcommand*{\GLSaccesssymbol}[1]{%
4976     \glssymbolaccessdisplay
4977     {%

```

```

4978     \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
4979   }%
4980   {#1}%
4981 }

```

`\glsaccesssymbolplural` Display the symbolplural value (no link and no check for existence).

```

4982 \newcommand*{\glsaccesssymbolplural}[1]{%
4983   \glsymbolpluralaccessdisplay
4984   {%
4985     \glsentrysymbolplural{#1}%
4986   }%
4987   {#1}%
4988 }

```

`\Glsaccesssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

4989 \newcommand*{\Glsaccesssymbolplural}[1]{%
4990   \glsymbolpluralaccessdisplay
4991   {%
4992     \Glsentrysymbolplural{#1}%
4993   }%
4994   {#1}%
4995 }

```

`\glsaccesssymbolplural` Display the symbolplural value (no link and no check for existence) converted to upper case.

```

4996 \newcommand*{\GLSaccesssymbolplural}[1]{%
4997   \glsymbolpluralaccessdisplay
4998   {%
4999     \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
5000   }%
5001   {#1}%
5002 }

```

`\glsaccessdesc` Display the desc value (no link and no check for existence).

```

5003 \newcommand*{\glsaccessdesc}[1]{%
5004   \glsdescriptionaccessdisplay
5005   {%
5006     \glsentrydesc{#1}%
5007   }%
5008   {#1}%
5009 }

```

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

5010 \newcommand*{\GLSaccessdesc}[1]{%
5011   \glsdescriptionaccessdisplay
5012   {%
5013     \Glsentrydesc{#1}%

```

```

5014     }%
5015     {#1}%
5016 }

```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```

5017 \newcommand*{\GLSaccessdesc}[1]{%
5018   \glsdescriptionaccessdisplay
5019   {%
5020     \mfirstucMakeUppercase{\glentrydesc{#1}}%
5021   }%
5022   {#1}%
5023 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence).

```

5024 \newcommand*{\glsaccessdescplural}[1]{%
5025   \glsdescriptionpluralaccessdisplay
5026   {%
5027     \glentrydescplural{#1}%
5028   }%
5029   {#1}%
5030 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

5031 \newcommand*{\Glsaccessdescplural}[1]{%
5032   \glsdescriptionpluralaccessdisplay
5033   {%
5034     \Glsentrydescplural{#1}%
5035   }%
5036   {#1}%
5037 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

5038 \newcommand*{\GLSaccessdescplural}[1]{%
5039   \glsdescriptionpluralaccessdisplay
5040   {%
5041     \mfirstucMakeUppercase{\glentrydescplural{#1}}%
5042   }%
5043   {#1}%
5044 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

5045 \newcommand*{\glsaccessshort}[1]{%
5046   \glsshortaccessdisplay
5047   {%
5048     \glentryshort{#1}%
5049   }%
5050   {#1}%
5051 }

```

`\Glsaccesssshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5052 \newcommand*{\Glsaccesssshort}[1]{%
5053   \glsshortaccessdisplay
5054   {%
5055     \Glsentryshort{#1}%
5056   }%
5057   {#1}%
5058 }

```

`\GLSaccesssshort` Display the short value (no link and no check for existence) converted to upper case.

```

5059 \newcommand*{\GLSaccesssshort}[1]{%
5060   \glsshortaccessdisplay
5061   {%
5062     \mfirstucMakeUppercase{\Glsentryshort{#1}}%
5063   }%
5064   {#1}%
5065 }

```

`laccesssshortpl` Display the short plural form (no link and no check for existence).

```

5066 \newcommand*{\laccesssshortpl}[1]{%
5067   \glsshortpluralaccessdisplay
5068   {%
5069     \Glsentryshortpl{#1}%
5070   }%
5071   {#1}%
5072 }

```

`laccesssshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5073 \newcommand*{\laccesssshortpl}[1]{%
5074   \glsshortpluralaccessdisplay
5075   {%
5076     \Glsentryshortpl{#1}%
5077   }%
5078   {#1}%
5079 }

```

`LSaccesssshortpl` Display the shortplural value (no link and no check for existence) converted to upper case.

```

5080 \newcommand*{\LSaccesssshortpl}[1]{%
5081   \glsshortpluralaccessdisplay
5082   {%
5083     \mfirstucMakeUppercase{\Glsentryshortpl{#1}}%
5084   }%
5085   {#1}%
5086 }

```

`\glaccesslong` Display the long form (no link and no check for existence).

```

5087 \newcommand*\glsaccesslong}[1]{%
5088   \glslongaccessdisplay{\glsentrylong{#1}}{#1}%
5089 }

```

`\Glsaccesslong` Display the long form (no link and no check for existence).

```

5090
5091 \newcommand*\Glsaccesslong}[1]{%
5092   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5093 }

```

`\GLSaccesslong` Display the long value (no link and no check for existence) converted to upper case.

```

5094 \newcommand*\GLSaccesslong}[1]{%
5095   \glslongaccessdisplay
5096   {%
5097     \mfirstucMakeUppercase{\glsentrylong{#1}}%
5098   }%
5099   {#1}%
5100 }

```

`\glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

5101 \newcommand*\glsaccesslongpl}[1]{%
5102   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
5103 }

```

`\Glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

5104
5105 \newcommand*\Glsaccesslongpl}[1]{%
5106   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5107 }

```

`\GLSaccesslongpl` Display the longplural value (no link and no check for existence) converted to upper case.

```

5108 \newcommand*\GLSaccesslongpl}[1]{%
5109   \glslongpluralaccessdisplay
5110   {%
5111     \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5112   }%
5113   {#1}%
5114 }

```

End of if part

```

5115 }
5116 {

```

No accessibility support. Just define these commands to do `\glsentry<xxx>`

`\glsaccessname` Display the name value (no link and no check for existence).

```

5117 \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.

5118 `\newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}`

`\GLSaccessname` Display the name value (no link and no check for existence). converted to upper case.

5119 `\newcommand*{\GLSaccessname}[1]{%`
5120 `\protect\mfirstucMakeUppercase{\glentryname{#1}}}`

`\glsaccesstext` Display the text value (no link and no check for existence).

5121 `\newcommand*{\glsaccesstext}[1]{\glentrytext{#1}}`

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

5122 `\newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}`

`\GLSaccesstext` Display the text value (no link and no check for existence). converted to upper case.

5123 `\newcommand*{\GLSaccesstext}[1]{%`
5124 `\protect\mfirstucMakeUppercase{\glentrytext{#1}}}`

`glsaccessplural` Display the plural value (no link and no check for existence).

5125 `\newcommand*{\glsaccessplural}[1]{\glentryplural{#1}}`

`GLsaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

5126 `\newcommand*{\GLsaccessplural}[1]{\GLsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.

5127 `\newcommand*{\GLSaccessplural}[1]{%`
5128 `\protect\mfirstucMakeUppercase{\glentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).

5129 `\newcommand*{\glsaccessfirst}[1]{\glentryfirst{#1}}`

`\GLsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.

5130 `\newcommand*{\GLsaccessfirst}[1]{\GLsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.

5131 `\newcommand*{\GLSaccessfirst}[1]{%`
5132 `\protect\mfirstucMakeUppercase{\glentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).

5133 `\newcommand*{\glsaccessfirstplural}[1]{\glentryfirstplural{#1}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

5134 `\newcommand*{\GLsaccessfirstplural}[1]{\GLsentryfirstplural{#1}}`

`GLSaccessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.
5135 `\newcommand*{\GLSaccessfirstplural}[1]{%`
5136 `\protect\mfirstucMakeUppercase{\glentryfirstplural{#1}}}`

`glsaccesssymbol` Display the symbol value (no link and no check for existence).
5137 `\newcommand*{\glsaccesssymbol}[1]{\glentrysymbol{#1}}`

`GLsaccesssymbol` Display the symbol value (no link and no check for existence) with the first letter converted to upper case.
5138 `\newcommand*{\GLsaccesssymbol}[1]{\GLentrysymbol{#1}}`

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.
5139 `\newcommand*{\GLSaccesssymbol}[1]{%`
5140 `\protect\mfirstucMakeUppercase{\glentrysymbol{#1}}}`

`GLSaccesssymbolplural` Display the symbolplural value (no link and no check for existence).
5141 `\newcommand*{\GLSaccesssymbolplural}[1]{\glentrysymbolplural{#1}}`

`GLsaccesssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.
5142 `\newcommand*{\GLsaccesssymbolplural}[1]{\GLentrysymbolplural{#1}}`

`GLSaccesssymbolplural` Display the symbolplural value (no link and no check for existence). converted to upper case.
5143 `\newcommand*{\GLSaccesssymbolplural}[1]{%`
5144 `\protect\mfirstucMakeUppercase{\glentrysymbolplural{#1}}}`

`\glsaccessdesc` Display the desc value (no link and no check for existence).
5145 `\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.
5146 `\newcommand*{\GLsaccessdesc}[1]{\GLentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence). converted to upper case.
5147 `\newcommand*{\GLSaccessdesc}[1]{%`
5148 `\protect\mfirstucMakeUppercase{\glentrydesc{#1}}}`

`GLSaccessdescplural` Display the descplural value (no link and no check for existence).
5149 `\newcommand*{\GLSaccessdescplural}[1]{\glentrydescplural{#1}}`

`GLsaccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.
5150 `\newcommand*{\GLsaccessdescplural}[1]{\GLentrydescplural{#1}}`

ccessdescplural Display the descplural value (no link and no check for existence). converted to upper case.

```

5151 \newcommand*{\GLSaccessdescplural}[1]{%
5152 \protect\mfirstucMakeUppercase{\glentrydescplural{#1}}}
```

\glsaccessshort Display the short form (no link and no check for existence).

```

5153 \newcommand*{\glsaccessshort}[1]{\glentryshort{#1}}
```

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5154 \newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}
```

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.

```

5155 \newcommand*{\GLSaccessshort}[1]{%
5156 \protect\mfirstucMakeUppercase{\glentryshort{#1}}}
```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```

5157 \newcommand*{\glsaccessshortpl}[1]{\glentryshortpl{#1}}
```

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5158 \newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.

```

5159 \newcommand*{\GLSaccessshortpl}[1]{%
5160 \protect\mfirstucMakeUppercase{\glentryshortpl{#1}}}
```

\glsaccesslong Display the long form (no link and no check for existence).

```

5161 \newcommand*{\glsaccesslong}[1]{\glentrylong{#1}}
```

\Glsaccesslong Display the long form (no link and no check for existence).

```

5162 \newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}
```

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.

```

5163 \newcommand*{\GLSaccesslong}[1]{%
5164 \protect\mfirstucMakeUppercase{\glentrylong{#1}}}
```

glsaccesslongpl Display the long plural form (no link and no check for existence).

```

5165 \newcommand*{\glsaccesslongpl}[1]{\glentrylongpl{#1}}
```

Glsaccesslongpl Display the long plural form (no link and no check for existence).

```

5166 \newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

GLSaccesslongpl Display the longplural value (no link and no check for existence). converted to upper case.

```

5167 \newcommand*{\GLSaccesslongpl}[1]{%
5168 \protect\mfirstucMakeUppercase{\glentrylongpl{#1}}}
```

End of else part

```

5169 }
```


1.6 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```
5170 \glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```
5171 \newcommand{\glsifcategory}[4]{%
5172   \ifglsfieldeq{#1}{category}{#2}{#3}{#4}%
5173 }
```

Categories can have attributes.

`categoryattribute`

```
\glssetcategoryattribute{<category>}{<attribute-label>}{<value>}
```

Set (or override if already set) an attribute for the given category.

```
5174 \newcommand*{\glssetcategoryattribute}[3]{%
5175   \csdef{@glxtr@categoryattr@@#1@#2}{#3}%
5176 }
```

`categoryattribute`

```
\glsgetcategoryattribute{<category>}{<attribute-label>}
```

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
5177 \newcommand*{\glsgetcategoryattribute}[2]{%
5178   \csuse{@glxtr@categoryattr@@#1@#2}%
5179 }
```

`categoryattribute`

```
\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}
```

Tests if the category has the given attribute set.

```
5180 \newcommand*{\glshascategoryattribute}[4]{%
5181   \ifcvoid{@glxtr@categoryattr@@#1@#2}{#4}{#3}%
5182 }
```

`\glssetattribute`

```
\glssetattribute{<entry label>}{<attribute-label>}{<value>}
```

Short cut where the category label is obtained from the entry information.

```
5183 \newcommand*{\glssetattribute}[3]{%
```

```

5184 \glsssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5185 }

```

`\glsggetattribute` `\glsggetattribute{<entry label>}{<attribute-label>}`

Short cut where the category label is obtained from the entry information.

```

5186 \newcommand*{\glsggetattribute}[2]{%
5187   \glsssetcategoryattribute{\glscategory{#1}}{#2}%
5188 }

```

`\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.

```

5189 \newcommand*{\glshasattribute}[4]{%
5190   \ifglssentryexists{#1}%
5191   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5192   {#4}%
5193 }

```

`categoryattribute` `\glssifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

True if category has the attribute with the given value.

```

5194 \newcommand{\glssifcategoryattribute}[5]{%
5195   \ifcsundef{@glssxtr@categoryattr@#1@#2}%
5196   {#5}%
5197   {\ifcsstring{@glssxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
5198 }

```

`\glssifattribute` `\glssifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

Short cut to determine if the given entry has a category with the given attribute set.

```

5199 \newcommand{\glssifattribute}[5]{%
5200   \ifglssentryexists{#1}%
5201   {\glssifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5202   {#5}%
5203 }

```

Set attributes for the default general category:

```
5204 \glssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
5205 \glssetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```
5206 \newcommand*\glssetregularcategory}[1]{%  
5207 \glssetcategoryattribute{#1}{regular}{true}}%  
5208 }
```

`ifregularcategory` `\glsifregularcategory{<category>}{<true part>}{<>false part>}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```
5209 \newcommand{\glsifregularcategory}[3]{%  
5210 \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}}%  
5211 }
```

`ifnotregularcategory` `\glsifnotregularcategory{<category>}{<true part>}{<>false part>}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```
5212 \newcommand{\glsifnotregularcategory}[3]{%  
5213 \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}}%  
5214 }
```

`\glsifregular` `\glsifregular{<entry label>}{<true part>}{<>false part>}`

Short cut to determine if an entry has a regular attribute set to true.

```
5215 \newcommand{\glsifregular}[3]{%  
5216 \glsifregularcategory{\glscategory{#1}}{#2}{#3}}%  
5217 }
```

`\glsifnotregular` `\glsifnotregular{<entry label>}{<true part>}{<>false part>}`

Short cut to determine if an entry has a regular attribute set to false.

```
5218 \newcommand{\glsifnotregular}[3]{%  
5219 \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}}%  
5220 }
```

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.

```
5221 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
5222   \forallglossaries[#1]{#3}%
5223   {%
5224     \forallsentries[#3]{#4}%
5225     {%
5226       \glsifcategory{#4}{#2}{#5}{}%
5227     }%
5228   }%
5229 }
```

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.

```
5230 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5231   \forallglossaries[#1]{#4}%
5232   {%
5233     \forallsentries[#4]{#5}%
5234     {%
5235       \glsifattribute{#5}{#2}{#3}{#6}{}%
5236     }%
5237   }%
5238 }
```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to `index` and add `\glstrpostdescription`.

```
5239 \ifdef\newterm
5240 {%
```

`\newterm`

```
5241   \renewcommand*{\newterm}[2][ ]{%
5242     \newglossaryentry{#2}%
5243     {type={index},category=index,name={#2},%
```

```

5244     description={\glxtrpostdescription\nopostdesc},#1}%
5245 }

```

Indexed terms are regular by default.

```

5246 \glsssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

5247 \newcommand*{\glxtrpostdescindex}{}

5248 }
5249 {}

```

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.

```

5250 \ifdef\printsymbols
5251 {%

```

glxtrnewsymbol Unlike \newterm, this has a separate argument for the label (since the symbol will likely contain commands).

```

5252 \newcommand*{\glxtrnewsymbol}[3] [] {%
5253     \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5254 }

```

Symbols are regular by default.

```

5255 \glsssetcategoryattribute{symbol}{regular}{true}

```

rpostdescsymbol

```

5256 \newcommand*{\glxtrpostdescsymbol}{}

5257 }
5258 {}

```

Similar for the numbers option.

```

5259 \ifdef\printnumbers
5260 {%

```

glxtrnewnumber

```

5261 \ifdef\printnumbers
5262 \newcommand*{\glxtrnewnumber}[3] [] {%
5263     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5264 }

```

Numbers are regular by default.

```

5265 \glsssetcategoryattribute{number}{regular}{true}

```

rpostdescnumber

```

5266 \newcommand*{\glxtrpostdescnumber}{}

```

```
5267 }
5268 {}
```

`\glstrsetcategory` Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
5269 \newcommand*{\glstrsetcategory}[2]{%
5270   \@for\@glstr@label:=#1\do
5271   {%
5272     \glsfieldxdef{\@glstr@label}{category}{#2}%
5273   }%
5274 }
```

`\glstrcategoryforall` Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```
5275 \newcommand*{\glstrcategoryforall}[2]{%
5276   \forallglossaries[#1]{\@glstr@type}{%
5277     \forallsentries[\@glstr@type]{\@glstr@label}%
5278     {%
5279       \glsfieldxdef{\@glstr@label}{category}{#2}%
5280     }%
5281   }%
5282 }
```

`\glstrfieldtitlecase` `\glstrfieldtitlecase{\langle label \rangle}{\langle field \rangle}`

Apply title casing to the contents of the given field.

```
5283 \newcommand*{\glstrfieldtitlecase}[2]{%
5284   \expandafter\glstrfieldtitlecasesecs\expandafter
5285   {\csname glo@glsdetoklabel{#1}@#2\endcsname}%
5286 }
```

`\glstrfieldtitlecasesecs` The command used by `\glstrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```
5287 \newcommand*{\glstrfieldtitlecasesecs}[1]{\xcapitalisewords{#1}}
```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

`\glossentrydesc` If the `glossdesc` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```
5288 \@ifpackageloaded{glossaries-accsupp}
5289 {
5290   \renewcommand*{\glossentrydesc}[1]{%
5291     \glsdoifexistsorwarn{#1}%
5292     {%
5293       \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```

5294 \glshasattribute{#1}{glossdescfont}%
5295 {%
5296 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
5297 \ifcsdef{\@glxtr@attrval}%
5298 {%
5299 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5300 }%
5301 {%
5302 \GlossariesExtraWarning{Unknown control sequence name
5303 '\@glxtr@attrval' supplied in glossdescfont attribute
5304 for entry '#1'. Ignoring}%
5305 \let\@glxtr@glossdescfont\@firstofone
5306 }%
5307 }%
5308 {\let\@glxtr@glossdescfont\@firstofone}%
5309 \glusifattribute{#1}{glossdesc}{firstuc}%
5310 {%
5311 \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5312 }%
5313 {%
5314 \glusifattribute{#1}{glossdesc}{title}%
5315 {%
5316 \@glxtr@do@titlecaps@warn
5317 \glsdescriptionaccessdisplay
5318 {%
5319 \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
5320 }%
5321 {#1}%
5322 }%
5323 {%
5324 \@glxtr@glossdescfont{\glaccessdesc{#1}}%
5325 }%
5326 }%
5327 }%
5328 }
5329 }
5330 {
5331 \renewcommand*{\glossentrydesc}[1]{%
5332 \glsdoifexistsorwarn{#1}%
5333 {%
5334 \glsetabbrvfmt{\glscategory{#1}}%
5335 \glshasattribute{#1}{glossdescfont}%
5336 {%
5337 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
5338 \ifcsdef{\@glxtr@attrval}%
5339 {%
5340 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5341 }%

```

```

5342      {%
5343      \GlossariesExtraWarning{Unknown control sequence name
5344      '\@glsxtr@attrval' supplied in glossdescfont attribute
5345      for entry '#1'. Ignoring}%
5346      \let\@glsxtr@glossdescfont\@firstofone
5347      }%
5348    }%
5349    {\let\@glsxtr@glossdescfont\@firstofone}%
5350    \glsifattribute{#1}{glossdesc}{firstuc}%
5351    {%
5352      \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
5353    }%
5354    {%
5355      \glsifattribute{#1}{glossdesc}{title}%
5356      {%
5357        \@glsxtr@do@titlecaps@warn
5358        \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
5359      }%
5360      {%
5361        \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
5362      }%
5363    }%
5364  }%
5365 }
5366 }

```

`\glossentryname` If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

5367 \@ifpackageloaded{glossaries-accsupp}
5368 {
5369   \renewcommand*{\glossentryname}[1]{%
5370     \@glsdoifexistsorwarn{#1}%
5371     {%
5372       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

5373   \glshasattribute{#1}{glossnamefont}%
5374   {%
5375     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
5376     \ifcsdef{\@glsxtr@attrval}%
5377     {%
5378       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
5379     }%
5380     {%
5381       \GlossariesExtraWarning{Unknown control sequence name
5382       '\@glsxtr@attrval' supplied in glossnamefont attribute
5383       for entry '#1'. Reverting to default \string\glsnamefont}%
5384       \let\@glsxtr@glossnamefont\glsnamefont
5385     }%
5386   }%

```



```

5387 {\let\@glsxtr@glossnamefont\glsnamefont}%
5388 \glsifattribute{#1}{glossname}{firstuc}%
5389 {%
5390   \glsnameaccessdisplay
5391   {%
5392     \@glsxtr@glossnamefont{\Glsentryname{#1}}%
5393   }%
5394   {#1}%
5395 }%
5396 {%
5397   \glsifattribute{#1}{glossname}{title}%
5398   {%
5399     \@glsxtr@do@titlecaps@warn
5400     \glsnameaccessdisplay
5401     {%
5402       \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
5403     }%
5404     {#1}%
5405   }%
5406   {%
5407     \glsifattribute{#1}{glossname}{uc}%
5408     {%
5409       \glsnameaccessdisplay
5410       {%

```

Hide the label from the upper-casing command.

```

5411       \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5412       \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
5413     }%
5414     {#1}%
5415   }%
5416   {%
5417     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
5418     \glsnameaccessdisplay
5419     {%
5420       \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
5421     }%
5422     {#1}%
5423   }%
5424   }%
5425 }%

```

Do post-name hook:

```

5426   \glsxtrpostnamehook{#1}%
5427 }%
5428 }
5429 }
5430 {
5431   \renewcommand*{\glossentryname}[1]{%
5432     \@glsdoifexistsorwarn{#1}%

```

```

5433 {%
5434     \glsetabbrvfmt{\glscategory{#1}}%
5435     \glshasattribute{#1}{glossnamefont}%
5436     {%
5437         \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
5438         \ifcsdef{\@glxtr@attrval}%
5439         {%
5440             \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5441         }%
5442         {%
5443             \GlossariesExtraWarning{Unknown control sequence name
5444             '\@glxtr@attrval' supplied in glossnamefont attribute
5445             for entry '#1'. Reverting to default \string\glnamefont}%
5446             \let\@glxtr@glossnamefont\glnamefont
5447         }%
5448     }%
5449     {\let\@glxtr@glossnamefont\glnamefont}%
5450     \gl@ifattribute{#1}{glossname}{firstuc}%
5451     {%
5452         \@glxtr@glossnamefont{\Glsentryname{#1}}%
5453     }%
5454     {%
5455         \gl@ifattribute{#1}{glossname}{title}%
5456         {%
5457             \@glxtr@do@titlecaps@warn
5458             \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
5459         }%
5460         {%
5461             \gl@ifattribute{#1}{glossname}{uc}%
5462             {%

```

Hide the label from the upper-casing command.

```

5463         \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
5464         \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5465     }%
5466     {%

```

This little trick is used by glossaries to allow the user to redefine \glnamefont to use \makefirstuc. Support it even though they can now use the firstuc attribute.

```

5467         \letcs{\glo@name}{glo@\glsetoklabel{#1}@name}%
5468         \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
5469     }%
5470 }%
5471 }%

```

Do post-name hook.

```

5472     \glxtrpostnamehook{#1}%
5473 }%
5474 }
5475 }

```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```
5476 \@ifpackageloaded{glossaries-accsupp}
5477 {
5478   \renewcommand*{\Glossentryname}[1]{%
5479     \@glsdoifexistsorwarn{#1}%
5480     {%
5481       \glsetabbrvfmt{\glscategory{#1}}%
5482       \glshasattribute{#1}{glossnamefont}%
5483       {%
5484         \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
5485         \ifcsdef{\@glstr@attrval}%
5486         {%
5487           \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
5488         }%
5489         {%
5490           \GlossariesExtraWarning{Unknown control sequence name
5491             '\@glstr@attrval' supplied in glossnamefont attribute
5492             for entry '#1'. Reverting to default \string\glnamefont}%
5493           \let\@glstr@glossnamefont\glnamefont
5494         }%
5495       }%
5496       {\let\@glstr@glossnamefont\glnamefont}%
5497       \glnameaccessdisplay
5498       {%
5499         \@glstr@glossnamefont{\Glsentryname{#1}}%
5500       }%
5501       {#1}%
```

Do post-name hook:

```
5502   \glstrpostnamehook{#1}%
5503   }%
5504 }
5505 }
5506 {
5507   \renewcommand*{\Glossentryname}[1]{%
5508     \@glsdoifexistsorwarn{#1}%
5509     {%
5510       \glsetabbrvfmt{\glscategory{#1}}%
5511       \glshasattribute{#1}{glossnamefont}%
5512       {%
5513         \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
5514         \ifcsdef{\@glstr@attrval}%
5515         {%
5516           \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
5517         }%
5518         {%
5519           \GlossariesExtraWarning{Unknown control sequence name
5520             '\@glstr@attrval' supplied in glossnamefont attribute
```

```

5521         for entry ‘#1’. Reverting to default \string\glsnamefont}%
5522         \let\@glsxtr@glossnamefont\glsnamefont
5523     }%
5524 }%
5525 {\let\@glsxtr@glossnamefont\glsnamefont}%
5526 \@glsxtr@glossnamefont{\Glsentryname{#1}}}%

```

Do post-name hook:

```

5527     \glsxtrpostnamehook{#1}%
5528 }%
5529 }
5530 }

```

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`xtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry’s label.

```

5531 \newcommand*{\glsxtrpostnamehook}[1]{%
5532   \let\@glsnumberformat\@glsxtr@defaultnumberformat
5533   \glsxtrdoautoindexname{#1}{indexname}%

```

Allow additional code regardless of category:

```

5534   \glsextrapostnamehook{#1}%

```

Allow categories to hook in here.

```

5535   \csuse{glsxtrpostname\glscategory{#1}}%
5536 }

```

`trapostnamehook`

```

5537 \newcommand*{\glsextrapostnamehook}[1]{}%

```

`etaccessdisplay`

```

5538 \@ifpackageloaded{glossaries-accsupp}
5539 {
5540   \newcommand*{\glsxtr@setaccessdisplay}[1]{%
5541     \ifcsdef{gls#1accessdisplay}%
5542     {\letcs\@glsxtr@accessdisplay{gls#1accessdisplay}}%
5543     {%

```

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.

```

5544     \edef\@gls@thisval{#1}%
5545     \@for\@gls@map:=\@gls@keymap\do{%
5546       \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
5547       \ifdefequal{\@this@key}{\@gls@thisval}%
5548       {%
5549         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
5550         \@endfortrue

```

```

5551     }%
5552     {}%
5553     }%
5554     \ifcsdef{gls\@gls@thisval accessdisplay}%
5555     {\letcs\@glxtr@accessdisplay{gls\@gls@thisval accessdisplay}}%
5556     {\let\@glxtr@accessdisplay\@firstoftwo}%
5557     }%
5558 }
5559 }
5560 {%
5561 \newcommand*{\glxtr@setaccessdisplay}[1]{%
5562 \let\@glxtr@accessdisplay\@firstoftwo}
5563 }

```

sentrynameother Provide a command that works like `\glossentryname` but accesses a different field (which must be supplied using its internal field label).

```

5564 \newrobustcmd*{\glossentrynameother}[2]{%
5565 \@glstdoifexistsorwarn{#1}%
5566 {%

```

Accessibility support:

```

5567 \glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

5568 \glsssetabbrfmt{\glscategory{#1}}%
5569 \glshasattribute{#1}{glossnamefont}%
5570 {%
5571 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossnamefont}}%
5572 \ifcsdef{\@glxtr@attrval}%
5573 {%
5574 \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5575 }%
5576 {%
5577 \GlossariesExtraWarning{Unknown control sequence name
5578 '\@glxtr@attrval' supplied in glossnamefont attribute
5579 for entry '#1'. Reverting to default \string\glsgnamefont}%
5580 \let\@glxtr@glossnamefont\glsgnamefont
5581 }%
5582 }%
5583 {\let\@glxtr@glossnamefont\glsgnamefont}%
5584 \glsgifattribute{#1}{glossname}{firstuc}%
5585 {%
5586 \@glxtr@accessdisplay
5587 {\@glxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
5588 {#1}%
5589 }%
5590 {%
5591 \glsgifattribute{#1}{glossname}{title}%
5592 {%
5593 \@glxtr@do@titlecaps@warn

```

```

5594      \@glsxtr@accessdisplay
5595      {\@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{#2}}}%
5596      {#1}%
5597  }%
5598  {%
5599      \glsifattribute{#1}{glossname}{uc}%
5600      {%
5601          \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
5602          \@glsxtr@accessdisplay
5603          {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
5604          {#1}%
5605      }%
5606      {%
5607          \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
5608          \@glsxtr@accessdisplay
5609          {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
5610          {#1}%
5611      }%
5612  }%
5613 }%

Do post-name hook.
5614      \glsxtrpostnamehook{#1}%
5615 }%
5616 }

```

`format@override` Determines if the `format` key should override the indexing attribute value.

```

5617 \newif\if@glsxtr@format@override
5618 \@glsxtr@format@overridefalse

```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

5619 \@ifpackageloaded{hyperref}
5620 {
5621     If hyperref's hyperindex option is on, then hyperref will automatically add \hyperpage, so
5622     don't add it.
5623     \ifHy@hyperindex
5624     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
5625         \@glsxtr@format@override true
5626         \appto\theindex{\let\glshypernumber\@firstofone}%
5627     }
5628     \else
5629     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
5630         \@glsxtr@format@override true
5631         \appto\theindex{\let\glshypernumber\hyperpage}%
5632     }
5633 }
5634 \fi

```

```

5632 }
5633 {
5634   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
5635     \@glsxtr@format@overridetrue
5636   }
5637 }
5638 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

5639 \newcommand*{\glxtrdoautoindexname}[2]{%
5640   \glshasattribute{#1}{#2}%
5641   {%

```

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.

```

5642   \@glsxtr@autoindex@setname{#1}%

```

If the attribute value is simply “true” don't add an encap, otherwise use the value as the encap.

```

5643   \protected@edef\@glsxtr@attrval{\glsggetattribute{#1}{#2}}%
5644   \if@glsxtr@format@override

5645     \ifx\@glsnumberformat\@glsxtr@defaultnumberformat
5646     \else
5647       \let\@glsxtr@attrval\@glsnumberformat
5648     \fi
5649   \fi
5650   \ifdefstring{\@glsxtr@attrval}{true}%
5651   {}%
5652   {\eappto\@glo@name{\@glsxtr@autoindex@encap\@glsxtr@attrval}}%
5653   \expandafter\glxtrautoindex\expandafter{\@glo@name}%
5654   }%
5655   {}%
5656 }

```

glxtrautoindex

```

5657 \newcommand*{\glxtrautoindex}{\index}

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

5658 \newcommand*{\@glsxtr@autoindex@setname}[1]{%
5659   \protected@edef\@glo@name{\glxtrautoindexentry{#1}}%
5660   \glxtrautoindexassignsort{\@glo@sort}{#1}%
5661   \@gls@checkmkidxchars\@glo@sort
5662   \@glsxtr@autoindex@doextra@esc\@glo@sort
5663   \epreto\@glo@name{\@glo@sort\@glsxtr@autoindex@at}%
5664 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

5665 \newcommand*{\glxtrautoindexentry}[1]{\string\glstryname{#1}}

```

trautoindexsort Used to assign the sort value when auto-indexing.

```
5666 \newcommand*{\glxtrautoindexassignsort}[2]{%
5667   \glslentryfield{#1}{#2}{sort}%
5668 }
```

dex@doextra@esc

```
5669 \newcommand*{\@glxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
5670   \ifx\@glxtr@autoindex@esc\@gl@quotechar
5671   \else
5672     \def\@gl@checkedmkidx{}%
5673     \edef\@glxtr@checkspch{%
5674       \noexpand\@glxtr@autoindex@escquote\expandonce{#1}%
5675       \noexpand\@empty\@glxtr@autoindex@esc\noexpand\@nnil
5676       \@glxtr@autoindex@esc\noexpand\@empty\noexpand\@glxtr@endescspch}%
5677     \@glxtr@checkspch
5678     \let#1\@gl@checkedmkidx\relax
5679   \fi
```

Escape actual character unless it has already been escaped.

```
5680   \ifx\@glxtr@autoindex@at\@gl@actualchar
5681   \else
5682     \def\@gl@checkedmkidx{}%
5683     \edef\@glxtr@checkspch{%
5684       \noexpand\@glxtr@autoindex@escat\expandonce{#1}%
5685       \noexpand\@empty\@glxtr@autoindex@at\noexpand\@nnil
5686       \@glxtr@autoindex@at\noexpand\@empty\noexpand\@glxtr@endescspch}%
5687     \@glxtr@checkspch
5688     \let#1\@gl@checkedmkidx\relax
5689   \fi
```

Escape level character unless it has already been escaped.

```
5690   \ifx\@glxtr@autoindex@level\@gl@levelchar
5691   \else
5692     \def\@gl@checkedmkidx{}%
5693     \edef\@glxtr@checkspch{%
5694       \noexpand\@glxtr@autoindex@esclevel\expandonce{#1}%
5695       \noexpand\@empty\@glxtr@autoindex@level\noexpand\@nnil
5696       \@glxtr@autoindex@level\noexpand\@empty\noexpand\@glxtr@endescspch}%
5697     \@glxtr@checkspch
5698     \let#1\@gl@checkedmkidx\relax
5699   \fi
```

Escape encap character unless it has already been escaped.

```
5700   \ifx\@glxtr@autoindex@encap\@gl@encapchar
5701   \else
5702     \def\@gl@checkedmkidx{}%
5703     \edef\@glxtr@checkspch{%
5704       \noexpand\@glxtr@autoindex@escencap\expandonce{#1}%
5705       \noexpand\@empty\@glxtr@autoindex@encap\noexpand\@nnil
```



```

5706      \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
5707      \@glsxtr@checkspch
5708      \let#1\@gls@checkedmkidx\relax
5709      \fi
5710 }

```

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`.

```
5711 \newcommand*{\@glsxtr@autoindex@at}{}

```

`trSetActualChar` Set the actual character.

```

5712 \newcommand*{\GlsXtrSetActualChar}[1]{%
5713   \gdef\@glsxtr@autoindex@at{#1}%
5714   \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
5715     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
5716   }%
5717 }
5718 \@onlypreamble\GlsXtrSetActualChar
5719 \makeatother
5720 \GlsXtrSetActualChar{@}
5721 \makeatletter

```

`autoindex@encap` Encap character for use with `\index`.

```
5722 \newcommand*{\@glsxtr@autoindex@encap}{}

```

`XtrSetEncapChar` Set the encap character.

```

5723 \newcommand*{\GlsXtrSetEncapChar}[1]{%
5724   \gdef\@glsxtr@autoindex@encap{#1}%
5725   \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
5726     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
5727   }%
5728 }
5729 \GlsXtrSetEncapChar{|}
5730 \@onlypreamble\GlsXtrSetEncapChar

```

`autoindex@level` Level character for use with `\index`.

```
5731 \newcommand*{\@glsxtr@autoindex@level}{}

```

`XtrSetLevelChar` Set the encap character.

```

5732 \newcommand*{\GlsXtrSetLevelChar}[1]{%
5733   \gdef\@glsxtr@autoindex@level{#1}%
5734   \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%
5735     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
5736   }%
5737 }
5738 \GlsXtrSetLevelChar{!}
5739 \@onlypreamble\GlsXtrSetLevelChar

```

r@autoindex@esc Escape character for use with \index.

```
5740 \newcommand*{\@glsxtr@autoindex@esc}{"
```

lsXtrSetEscChar Set the escape character.

```
5741 \newcommand*{\GlsXtrSetEscChar}[1]{%
5742   \gdef\@glsxtr@autoindex@esc{#1}%
5743   \def\@glsxtr@autoindex@escquote##1##2##3\@glsxtr@endescspch{%
5744     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
5745   }%
5746 }
5747 \GlsXtrSetEscChar{"}
5748 \@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:

```
5749 \ifdef\actualchar
5750 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
5751 {}
```

Quote character \quotechar:

```
5752 \ifdef\quotechar
5753 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
5754 {}
```

Level character \levelchar:

```
5755 \ifdef\levelchar
5756 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
5757 {}
```

Encap character \encapchar:

```
5758 \ifdef\encapchar
5759 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
5760 {}
```

leto@endescspch

```
5761 \def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

toindex@esc@spch

<code>\@glsxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}</code>

```
5762 \newcommand*{\@glsxtr@autoindex@escspch}[5]{%
5763   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
5764   \toks@={#3}%
5765   \ifx\@nnil#3\relax
5766     \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
5767   \else
5768     \ifx\@nnil#4\relax
5769       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
5770     \def\@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
```

```

5771      #4#5\@glxtr@endescspch}%
5772  \else
5773    \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
5774    \@glxtr@autoindex@esc#1}%
5775    \def\@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
5776  \fi
5777 \fi
5778 \@glxtr@checkspch
5779 }

```

\Glossentrydesc Redefine to set the abbreviation format and accessibility support.

```

5780 \renewcommand*{\Glossentrydesc}[1]{%
5781   \glsdoifexistsorwarn{#1}%
5782   {%
5783     \glssetabbrvfmt{\glscategory{#1}}%
5784     \Glsaccessdesc{#1}%
5785   }%
5786 }

```

\Glossentrysymbol Redefine to set the abbreviation format and accessibility support.

```

5787 \renewcommand*{\Glossentrysymbol}[1]{%
5788   \glsdoifexistsorwarn{#1}%
5789   {%
5790     \glssetabbrvfmt{\glscategory{#1}}%
5791     \Glsaccesssymbol{#1}%
5792   }%
5793 }

```

\Glossentrysymbol Redefine to set the abbreviation format and accessibility support.

```

5794 \renewcommand*{\Glossentrysymbol}[1]{%
5795   \glsdoifexistsorwarn{#1}%
5796   {%
5797     \glssetabbrvfmt{\glscategory{#1}}%
5798     \Glsaccesssymbol{#1}%
5799   }%
5800 }

```

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.

```

5801 \newcommand*{\GlsXtrEnableInitialTagging}{%
5802   \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
5803 }
5804 \@onlypreamble\GlsXtrEnableInitialTagging

```

\GlsXtrEnableInitialTagging Starred version undefines command.

```

5805 \newcommand*{\s@glxstr@enabletagging}[2]{%
5806   \undef#2%
5807   \@glxstr@enabletagging{#1}{#2}%
5808 }

```

r@enabletagging Internal command.

```

5809 \newcommand*{\@glxstr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
5810   \@for\@glxstr@cat:=#1\do
5811   {%
5812     \ifdefempty\@glxstr@cat
5813     {}%
5814     {\glsssetcategoryattribute{\@glxstr@cat}{tagging}{true}}%
5815   }%
5816   \newrobustcmd*#2[1]{##1}%
5817   \def\@glxstr@taggingcs{#2}%
5818   \renewcommand*{\@glxstr@activate@initialtagging{%
5819     \let#2\@glxstr@tag
5820   }%
5821   \ifundef\@glspreglossaryhook
5822   {\GlossariesExtraWarning{Initial tagging requires at least
5823     glossaries.sty v4.19 to work correctly}}%
5824   {}%
5825 }

```

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

```

5826 \ifundef\mfu@checkword@do
5827 {
5828   \newcommand*{\mfu@checkword@do}[1]{%
5829     \ifdefstring{\mfu@checkword@arg}{#1}%
5830     {%
5831       \let\@mfu@domakefirstuc\@firstofone
5832       \listbreak
5833     }%
5834     {}%
5835   }

```

\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.

```

5836 \ifundef\mfu@checkword
5837 {
5838   \newcommand{\@glxstr@do@titlecaps@warn}{%
5839     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
5840       support not available}%

```

One warning should suffice.

```
5841 \let\@glxtr@do@titlecaps@warn\relax
5842 }
5843 }
5844 {
5845 \renewcommand*{\mfu@checkword}[1]{%
5846 \def\mfu@checkword@arg{#1}%
5847 \let\@mfu@domakefirstuc\makefirstuc
5848 \forlistloop\mfu@checkword@do\@mfu@nocaplist
5849 }
5850 }
5851 }
5852 {}% no patch required
```

@titlecaps@warn Do warning if title case not supported.

```
5853 \newcommand*{\@glxtr@do@titlecaps@warn}{}%
```

@initialtagging Used in \printglossary but at least v4.19 of glossaries required.

```
5854 \newcommand*{\@glxtr@activate@initialtagging}{}%
```

\@glxtr@tag Definition of tagging command when used in glossary.

```
5855 \newrobustcmd*{\@glxtr@tag}[1]{%
5856 \glsifattribute{\glscurrententrylabel}{tagging}{true}%
5857 {\glxtrtagfont{#1}}{#1}%
5858 }
```

\glxtrtagfont Used in the glossary.

```
5859 \newcommand*{\glxtrtagfont}[1]{\underline{#1}}
```

preglossaryhook This macro was introduced in glossaries version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.

```
5860 \ifdef\@gls@preglossaryhook
5861 {
5862 \renewcommand*{\@gls@preglossaryhook}{%
5863 \@glxtr@activate@initialtagging
```

Since the glossaries are automatically scoped, \@glxtr@org@postdescription shouldn't already be defined, but check anyway just as a precautionary measure.

```
5864 \ifundef\@glxtr@org@postdescription
5865 {%
5866 \let\@glxtr@org@postdescription\glspostdescription
5867 \renewcommand*{\glspostdescription}{%
5868 \ifglentryexists{\glscurrententrylabel}%
5869 {%
5870 \glxtrpostdescription
5871 \@glxtr@org@postdescription
```

```

5872     }%
5873     {}%
5874     }%
5875     }%
5876     {}%

```

Enable the options used by \@glsxtrp:

```

5877     \glossxtrsetpopts
5878     }%
5879 }
5880 {}

```

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.

```

5881 \newcommand*{\glsxtrpostdescription}{%
5882   \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
5883 }

```

postdescgeneral

```

5884 \newcommand*{\glsxtrpostdescgeneral}{}

```

xtrpostdescterm

```

5885 \newcommand*{\glsxtrpostdescterm}{}

```

postdescacronym

```

5886 \newcommand*{\glsxtrpostdescacronym}{}

```

escabbreviation

```

5887 \newcommand*{\glsxtrpostdescabbreviation}{}

```

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.

```

5888 \renewcommand*{\glspostlinkhook}{%
5889   \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
5890 }

```

xtrpostlinkhook The entry label should already be stored in \glslabel by \@gls@link.

```

5891 \newcommand*{\glsxtrpostlinkhook}{%
5892   \glsxtrdiscardperiod{\glslabel}%
5893   {\glsxtrpostlinkendsentence}%
5894   {\glsxtrifcustomdiscardperiod
5895     {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}}%
5896   {\glsxtrpostlink}%
5897   }%
5898 }

```

`omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
5899 \newcommand*{\glxtrifcustomdiscardperiod}[2]{#2}
```

`\glxtrpostlink`

```
5900 \newcommand*{\glxtrpostlink}{%
5901 \csuse{\glxtrpostlink\glscategory{\glslabel}}%
5902 }
```

`linkendsentence` Done by `\glxtrpostlinkhook` if a full stop is discarded.

```
5903 \newcommand*{\glxtrpostlinkendsentence}{%
5904 \ifcsdef{\glxtrpostlink\glscategory{\glslabel}}
5905 {%
5906 \csuse{\glxtrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
5907 .\spacefactor\sfcode'\. \relax
5908 }%
5909 {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
5910 \spacefactor\sfcode'\. \relax
5911 }%
5912 }
```

`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.

```
5913 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
5914 \glxtrifwasfirstuse{\space\glxtrparen{\glssaccessdesc{\glslabel}}}{}%
5915 }
```

`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.

```
5916 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
5917 \glxtrifwasfirstuse
5918 {%
5919 \ifglshassymbol{\glslabel}%
5920 {\space\glxtrparen{\glssaccesssymbol{\glslabel}}}%
5921 {}%
5922 }%
5923 {}%
5924 }
```

`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).

```

5925 \newcommand*{\glxtrdiscardperiod}[3]{%
5926   \glxtrifwasfirstuse
5927   {%
5928     \glusifattribute{#1}{retainfirstuseperiod}{true}%
5929     {#3}%
5930     {%
5931       \glusifattribute{#1}{discardperiod}{true}%
5932       {%
5933         \glusifplural
5934         {%
5935           \glusifattribute{#1}{pluraldiscardperiod}{true}%
5936           {\glxtrifperiod{#2}{#3}}%
5937           {#3}%
5938         }%
5939         {%
5940           \glxtrifperiod{#2}{#3}%
5941         }%
5942       }%
5943     }%
5944   }%
5945 }%
5946 {%
5947   \glusifattribute{#1}{discardperiod}{true}%
5948   {%
5949     \glusifplural
5950     {%
5951       \glusifattribute{#1}{pluraldiscardperiod}{true}%
5952       {\glxtrifperiod{#2}{#3}}%
5953       {#3}%
5954     }%
5955     {%
5956       \glxtrifperiod{#2}{#3}%
5957     }%
5958   }%
5959   {#3}%
5960 }%
5961 }

```

\glxtrifperiod Make a convenient user command to check if the next character is a full stop (period). Works like \@ifstar but uses \new@ifnextchar rather than \@ifnextchar

```

5962 \newcommand*{\glxtrifperiod}[1]{\new@ifnextchar.\{\@firstoftwo{#1}\}}

```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

glxtr@punclist List of characters identified as punctuation marks. (Be careful of babel shorthands!) This doesn't allow for punctuation marks made up from multiple characters (such as ' ').

```

5963 \newcommand*{\glxtr@punclist}{.,:;?!}

```

punctuationmark Add character to punctuation list.

```

5964 \newcommand*{\glxtraddpunctuationmark}[1]{\appto\glxtr@punclist{#1}}

```


unctuationmarks Reset the punctuation list.

```
5965 \newcommand*{\glxtrsetpunctuationmarks}[1]{\def\glxtr@punclist{#1}}
```

```
\glxtrifpunc    \glxtrifnextpunc{<true part>}{<false part>}
```

Test if this is followed by a punctuation mark. (Adapted from \new@ifnextchar.)

```
5966 \newcommand*{\glxtrifnextpunc}[2]{%
5967   \def\reserved@a{#1}%
5968   \def\reserved@b{#2}%
5969   \futurelet\@glspunc@token\glxtr@ifnextpunc
5970 }
```

glxtr@ifnextpunc

```
5971 \newcommand*{\glxtr@ifnextpunc}{%
5972   \glxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
5973   \reserved@b
5974 }
```

glxtr@ifpunctoken Test if the token given in the first argument is in the punctuation list.

```
5975 \newcommand*{\glxtr@ifpunctoken}[1]{%
5976   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
5977 }
```

glxtr@ifpunctoken

```
5978 \def\@glxtr@ifpunctoken#1#2{%
5979   \let\reserved@d=#2%
5980   \ifx\reserved@d\@nnil
5981     \let\glxtr@next\@glxtr@notfoundinlist
5982   \else
5983     \ifx#1\reserved@d
5984       \let\glxtr@next\@glxtr@foundinlist
5985     \else
5986       \let\glxtr@next\@glxtr@ifpunctoken
5987     \fi
5988   \fi
5989   \glxtr@next#1%
5990 }
```

glxtr@foundinlist

```
5991 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
5992 \def\@glxtr@notfoundinlist#1{\@secondoftwo}
```

glsxtrdopostpunc

```
\glsxtrdopostpunc{<code>}
```

If this is followed by a punctuation character, do <code> after the character otherwise do <code> before whatever comes next.

```
5993 \newcommand{\glsxtrdopostpunc}[1]{%
5994   \glsxtrifnextpunc{\@glsxtr@swaptwo{#1}}{#1}%
5995 }
```

@glsxtr@swaptwo

```
5996 \newcommand{\@glsxtr@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.

```
5997 \define@key{glsxtrabbrv}{category}{%
5998   \edef\glscategorylabel{#1}%
5999   \ifcsdef{@glsabbrv@current@#1}%
6000   {%
```

Warning should already have been issued.

```
6001   \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6002   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6003   \glsxtr@applyabbrvstyle{\csgname @glsabbrv@current@#1\endcsname}%
6004   \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
6005 }%
6006 {}%
6007 }
```

Save the short plural form. This may be needed before the entry is defined.

```
6008 \define@key{glsxtrabbrv}{shortplural}{%
6009   \def\@gls@shortpl{#1}%
6010 }
```

Similarly for the long plural form.

```
6011 \define@key{glsxtrabbrv}{longplural}{%
6012   \def\@gls@longpl{#1}%
6013 }
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```
6014 \newtoks\glsshortpltok
```

```

\glslongpltok
6015 \newtoks\glslongpltok

\glsxtr@insertdots Provided in case user wants to automatically insert dots between each letter of the abbrevi-
ation. 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.
6016 \newcommand*{\@glsxtr@insertdots}[2]{%
6017   \def#1{}%
6018   \@glsxtr@insert@dots#1#2\@nnil
6019 }

\glsxtr@insert@dots
6020 \newcommand*{\@glsxtr@insert@dots}[2]{%
6021   \ifx\@nnil#2\relax
6022   \let\@glsxtr@insert@dots@next\@gobble
6023   \else
6024   \ifx\relax#2\relax
6025   \else
6026     \appto#1{#2.}%
6027   \fi
6028   \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
6029   \fi
6030   \@glsxtr@insert@dots@next#1%
6031 }

Similarly provide a way of replacing spaces with \glsxtrwordsep, which first needs to be
defined:

\glsxtrwordsep
6032 \newcommand*{\glsxtrwordsep}{\space}

Each word is marked with

\glsxtrword
6033 \newcommand*{\glsxtrword}[1]{#1}

\glsxtr@markwordseps
6034 \newcommand*{\@glsxtr@markwordseps}[2]{%
6035   \def#1{}%
6036   \@glsxtr@mark@wordseps#1#2 \@nnil
6037 }

\glsxtr@mark@wordseps
6038 \def\@glsxtr@mark@wordseps#1#2 #3{%
6039   \ifdefempty{#1}%

```

```

6040 {\def#1{\protect\glxstrword{#2}}}%
6041 {\appto#1{\protect\glxstrwordsep\protect\glxstrword{#2}}}%
6042 \ifx\@nnil#3\relax
6043 \let\@glxstr@mark@wordseps@next\relax
6044 \else
6045 \def\@glxstr@mark@wordseps@next{%
6046 \@glxstr@mark@wordseps#1#3}%
6047 \fi
6048 \@glxstr@mark@wordseps@next
6049 }

```

newabbreviation Define a new generic abbreviation.

```

6050 \newcommand*{\newabbreviation}[4][]{%
6051 \glxstr@newabbreviation{#1}{#2}{#3}{#4}%
6052 }

```

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.)

```

6053 \newcommand*{\glxstr@newabbreviation}[4]{%
6054 \glscopylisttok{#1}%
6055 \glscopylabeltok{#2}%
6056 \glscopyshorttok{#3}%
6057 \glscopylongtok{#4}%

```

Save the original short and long values (before attribute settings modify them).

```

6058 \def\glxstr@orgshort{#3}%
6059 \def\glxstr@orglong{#4}%

```

Get the category.

```

6060 \def\glscopycategorylabel{abbreviation}%
6061 \glxstr@applyabbrvstyle{\@glscopyabbrv@current@abbreviation}%

```

Ignore the shortplural and longplural keys.

```

6062 \setkeys*{\glxstr@abbrv}{shortplural,longplural}{#1}%

```

Set the default long plural

```

6063 \def\@glscopylongpl{#4\glscopypluralsuffix}%
6064 \let\@glscopydefault@longpl\@glscopylongpl

```

Has the markwords attribute been set?

```

6065 \glscopyifcategoryattribute{\glscopycategorylabel}{markwords}{true}%
6066 {%
6067 \@glscopytr@markwordseps\@glscopylong{#4}%
6068 \expandafter\def\expandafter\@glscopylongpl\expandafter
6069 {\@glscopylong\glscopypluralsuffix}%
6070 \let\@glscopydefault@longpl\@glscopylongpl

```

Update \glscopylongtok.

```

6071 \expandafter\glscopylongtok\expandafter{\@glscopylong}%
6072 }%
6073 {}%

```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```
6074 \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
6075 {%
6076   \@glstr@markwordseps\@gls@short{#3}%
6077 }%
6078 {%
```

Has the insertdots attribute been set?

```
6079 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
6080 {%
6081   \@glstr@insertdots\@gls@short{#3}%
6082   \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
6083 }%
6084 {\def\@gls@short{#3}}%
6085 }%
```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```
6086 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
6087 {%
6088   \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6089     '\abbrvpluralsuffix}%
6090 }%
6091 {%
```

Has the noshortplural attribute been set?

```
6092 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
6093 {%
6094   \let\@gls@shortpl\@gls@short
6095 }%
6096 {%
6097   \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
6098     \abbrvpluralsuffix}%
6099 }%
6100 }%
```

Update \glsshorttok:

```
6101 \expandafter\glsshorttok\expandafter{\@gls@short}%
```

Hook for further customisation if required:

```
6102 \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).

```
6103 \setkeys*{glstrabbrv}[category]{#1}%
```

Has the plural been explicitly set?

```
6104 \ifx\@gls@default@longpl\@gls@longpl
6105 \else
```

Has the markwords attribute been set?

```
6106 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
6107 {%
```

```

6108      \expandafter\@glxtr@markwordseps\expandafter\@gls@longpl\expandafter
6109      {\@gls@longpl}%
6110      }%
6111      {}%
6112      \fi

```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```

6113      \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
6114      \expandafter\glslongpltok\expandafter{\@gls@longpl}%

```

Do any extra setup provided by hook:

```

6115      \newabbreviationhook

```

Define this entry:

```

6116      \protected@edef\@do@newglossaryentry{%
6117      \noexpand\newglossaryentry{\the\glslabeltok}%
6118      {%
6119      type=\glxtrabbrvtype,%
6120      category=abbreviation,%
6121      short={\the\glsshorttok},%
6122      shortplural={\the\glsshortpltok},%
6123      long={\the\glslongtok},%
6124      longplural={\the\glslongpltok},%
6125      name={\the\glsshorttok},%
6126      \CustomAbbreviationFields,%
6127      \the\glskeylisttok
6128      }%
6129      }%
6130      \@do@newglossaryentry
6131      \GlsXtrPostNewAbbreviation
6132      }

```

`evpresetkeyhook` Hook for extra stuff in `\newabbreviation`

```

6133 \newcommand*{\glxtrnewabbrevpresetkeyhook}[3]{}

```

`NewAbbreviation` Hook used by abbreviation styles.

```

6134 \newcommand*{\GlsXtrPostNewAbbreviation}{}

```

`bbreviationhook` Hook for use with `\newabbreviation`.

```

6135 \newcommand*{\newabbreviationhook}{}

```

`reviationFields`

```

6136 \newcommand*{\CustomAbbreviationFields}{}

```

`\glxtrparen` For the parenthetical styles.

```

6137 \newcommand*{\glxtrparen}[1]{(#1)}

```

`lsxtrfullformat` Full format without case change.

```

6138 \newcommand*{\glxtrfullformat}[2]{%

```

```

6139 \glsfirstlongfont{\glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
6140 \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
6141 }

```

`\glsxtrfullformat` Full format with case change.

```

6142 \newcommand*{\Glsxtrfullformat}[2]{%
6143 \glsfirstlongfont{\Glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
6144 \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{#1}}}%
6145 }

```

`\glsxtrfullplformat` Plural full format without case change.

```

6146 \newcommand*{\glsxtrfullplformat}[2]{%
6147 \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
6148 \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6149 }

```

`\glsxtrfullplformat` Plural full format with case change.

```

6150 \newcommand*{\Glsxtrfullplformat}[2]{%
6151 \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
6152 \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
6153 }

```

`\glsxtrfullsep` Separator used by full format is a space by default. The argument is the entry's label.

```

6154 \newcommand*{\glsxtrfullsep}[1]{\space}

```

In-line formats in case first use isn't compatible with `\glsentryfull` (for example, first use suppresses the long form or uses a footnote).

`\glsxtrinfullformat` Full format without case change.

```

6155 \newcommand*{\glsxtrinfullformat}{\glsxtrfullformat}

```

`\glsxtrinfullformat` Full format with case change.

```

6156 \newcommand*{\Glsxtrinfullformat}{\Glsxtrfullformat}

```

`\glsxtrfullplformat` Plural full format without case change.

```

6157 \newcommand*{\glsxtrinfullplformat}{\glsxtrfullplformat}

```

`\glsxtrinfullplformat` Plural full format with case change.

```

6158 \newcommand*{\Glsxtrinfullplformat}{\Glsxtrfullplformat}

```

Redefine `\glsentryfull` etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the `\glsxtrfull` set of commands instead.

`\glsentryfull`

```

6159 \renewcommand*{\glsentryfull}[1]{\glsxtrinfullformat{#1}{}}

```

`\Glsentryfull`

```

6160 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinfullformat{#1}{}}

```

`\glsentryfullpl`

```
6161 \renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}
```

`\Glsentryfullpl`

```
6162 \renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}
```

`\glsfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.

```
6163 \newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}
```

`\glsabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.

```
6164 \newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}
```

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.

```
6165 \newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}
```

`\glsabbrvdefaultfont`

```
6166 \newcommand*{\glsabbrvdefaultfont}[1]{#1}
```

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.

```
6167 \newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}
```

`\glslongdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.

```
6168 \newcommand*{\glslongdefaultfont}[1]{#1}
```

`\glsfirstlongfont` Font changing command used for the long form on first use or in the full format.

```
6169 \newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}
```

`\glsfirstlongdefaultfont`

```
6170 \newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}
```

`\glsbrvppluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.

```
6171 \newcommand*{\glsxtrabbrvppluralsuffix}{\glspluralsuffix}
```

`\glsbrvppluralsuffix` Default plural suffix.

```
6172 \newcommand*{\abbrvppluralsuffix}{\glsxtrabbrvppluralsuffix}
```

`\glsxtrfull` Full form (no case-change).

```
6173 \newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}
6174 \newcommand*{\ns@glsxtrfull}[2][{}]{%
6175   \new@ifnextchar[{\@glsxtr@full{#1}{#2}}{%
6176     {\@glsxtr@full{#1}{#2}}[{}]}%
6177 }
```


\@glsxtr@full Low-level macro:

```
6178 \def\@glsxtr@full#1#2[#3]{%
6179   \glsdoifexists{#2}%
6180   {%
6181     \glssetabbrvfmt{\glscategory{#2}}%
6182     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6183     \let\glsifplural\@secondoftwo
6184     \let\glsupcase\@firstofthree
6185     \let\glsinsert\@empty
6186     \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.

```
6187   \glsxtrsetupfulldefs
6188   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6189   }%
6190   \glspostlinkhook
6191 }
```

trsetupfulldefs

```
6192 \newcommand*\@glsxtrsetupfulldefs{%
6193   \let\glsxtrifwasfirstuse\@firstoftwo
6194 }
```

\Glsxtrfull Full form (first letter uppercase).

```
6195 \newrobustcmd*\Glsxtrfull{\@gls@hyp@opt\@ns@Glsxtrfull}
6196 \newcommand*\ns@Glsxtrfull[2][{}]{%
6197   \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
6198   {\@Glsxtr@full{#1}{#2}[]}%
6199 }
```

\@Glsxtr@full Low-level macro:

```
6200 \def\@Glsxtr@full#1#2[#3]{%
6201   \glsdoifexists{#2}%
6202   {%
6203     \glssetabbrvfmt{\glscategory{#2}}%
6204     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6205     \let\glsifplural\@secondoftwo
6206     \let\glsupcase\@secondofthree
6207     \let\glsinsert\@empty
6208     \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6209     \glsxtrsetupfulldefs
6210     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6211     }%
6212     \glspostlinkhook
6213 }
```

`\GLSxtrfull` Full form (all uppercase).

```
6214 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
6215 \newcommand*\ns@GLSxtrfull[2][\%
6216 \new@ifnextchar[\@GLSxtr@full{#1}{#2}]{%
6217 {\@GLSxtr@full{#1}{#2}[]}%
6218 }
```

`\@GLSxtr@full` Low-level macro:

```
6219 \def\@GLSxtr@full#1#2[#3]{%
6220 \glsdoifexists{#2}%
6221 {%
6222 \glssetabbrvfmt{\glscategory{#2}}%
6223 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6224 \let\glsifplural\@secondoftwo
6225 \let\glsapscase\@thirdofthree
6226 \let\glsinsert\@empty
6227 \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
6228 \glsxtrsetupfulldefs
6229 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6230 }%
6231 \glspostlinkhook
6232 }
```

`\glsxtrfullpl` Plural full form (no case-change).

```
6233 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\ns@glsxtrfullpl}
6234 \newcommand*\ns@glsxtrfullpl[2][\%
6235 \new@ifnextchar[\@glsxtr@fullpl{#1}{#2}]{%
6236 {\@glsxtr@fullpl{#1}{#2}[]}%
6237 }
```

`\@glsxtr@fullpl` Low-level macro:

```
6238 \def\@glsxtr@fullpl#1#2[#3]{%
6239 \glsdoifexists{#2}%
6240 {%
6241 \glssetabbrvfmt{\glscategory{#2}}%
6242 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6243 \let\glsifplural\@firstoftwo
6244 \let\glsapscase\@firstofthree
6245 \let\glsinsert\@empty
6246 \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
6247 \glsxtrsetupfulldefs
6248 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6249 }%
6250 \glspostlinkhook
6251 }
```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```
6252 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\ns@Glsxtrfullpl}
6253 \newcommand*\ns@Glsxtrfullpl[2][\%
```

```

6254 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
6255           {\@Glsxtr@fullpl{#1}{#2}[]}%
6256 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

6257 \def\@Glsxtr@fullpl#1#2[#3]{%
6258   \glsdoifexists{#2}%
6259   {%
6260     \glssetabbrvfmt{\glscategory{#2}}%
6261     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6262     \let\glsifplural\@firstoftwo
6263     \let\glsupcase\@secondofthree
6264     \let\glsinsert\@empty
6265     \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
6266     \glsxtrsetupfulldefs
6267     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6268   }%
6269   \glspostlinkhook
6270 }

```

`\Glsxtrfullpl` Plural full form (all upper case).

```

6271 \newrobustcmd*{\Glsxtrfullpl}{\@gl@hyp@opt\@ns@Glsxtrfullpl}
6272 \newcommand*\ns@Glsxtrfullpl[2][]{%
6273   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
6274           {\@Glsxtr@fullpl{#1}{#2}[]}%
6275 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

6276 \def\@Glsxtr@fullpl#1#2[#3]{%
6277   \glsdoifexists{#2}%
6278   {%
6279     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6280     \let\glsifplural\@firstoftwo
6281     \let\glsupcase\@thirdofthree
6282     \let\glsinsert\@empty
6283     \def\glscustomtext{%
6284       \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
6285     \glsxtrsetupfulldefs
6286     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6287   }%
6288   \glspostlinkhook
6289 }

```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```

6290 \newrobustcmd*{\glsxtrshort}{\@gl@hyp@opt\@ns@glsxtrshort}

```

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

```

6291 \newcommand*{\ns@glstrshort}[2] [] {%
6292   \new@ifnextchar[{\@glstrshort{#1}{#2}}{\@glstrshort{#1}{#2} []}%
6293 }

```

Read in the final optional argument:

```

6294 \def\@glstrshort#1#2[#3] {%
6295   \glstoifexists{#2}%
6296   {%

```

Need to make sure \glabbrvfont is set correctly.

```

6297   \glsetabbrvfmt{\glscategory{#2}}%
6298   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6299   \let\glstrifwasfirstuse\@secondoftwo
6300   \let\gl@ifplural\@secondoftwo
6301   \let\glscapscase\@firstofthree
6302   \let\glinsert\@empty
6303   \def\glscustomtext{%
6304     \glabbrvfont{\glaccessshort{#2}\ifglstrinsertinside#3\fi}%
6305     \ifglstrinsertinside\else#3\fi
6306   }%
6307   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6308 }%
6309 \glspostlinkhook
6310 }

```

\Glsxtrshort

```

6311 \newrobustcmd*{\Glsxtrshort}{\@gl@hyp@opt\ns@Glsxtrshort}

```

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

```

6312 \newcommand*{\ns@Glsxtrshort}[2] [] {%
6313   \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
6314 }

```

Read in the final optional argument:

```

6315 \def\@Glsxtrshort#1#2[#3] {%
6316   \glstoifexists{#2}%
6317   {%
6318     \glsetabbrvfmt{\glscategory{#2}}%
6319     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6320     \let\glstrifwasfirstuse\@secondoftwo
6321     \let\gl@ifplural\@secondoftwo
6322     \let\glscapscase\@secondofthree
6323     \let\glinsert\@empty
6324     \def\glscustomtext{%
6325       \glabbrvfont{\Glsaccessshort{#2}\ifglstrinsertinside#3\fi}%
6326       \ifglstrinsertinside\else#3\fi
6327     }%
6328     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6329   }%
6330   \glspostlinkhook
6331 }

```

\GLSxtrshort

```
6332 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\@ns@GLSxtrshort}
```

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

```
6333 \newcommand*{\ns@GLSxtrshort}[2][\%  
6334 \new@ifnextchar[\@GLSxtrshort{#1}{#2}]{\@GLSxtrshort{#1}{#2}[]}%  
6335 }
```

Read in the final optional argument:

```
6336 \def\@GLSxtrshort#1#2[#3]{%  
6337 \glsdoifexists{#2}%  
6338 {%  
6339 \glssetabbrvfmt{\glscategory{#2}}%  
6340 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
6341 \let\glsxtrifwasfirstuse\@secondoftwo  
6342 \let\glsifplural\@secondoftwo  
6343 \let\glsapscase\@thirdofthree  
6344 \let\glsinsert\@empty  
6345 \def\glscustomtext{%  
6346 \mfirstucMakeUppercase  
6347 {\glsabbrvfont{\glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%  
6348 \ifglsxtrininsertinside\else#3\fi  
6349 }%  
6350 }%  
6351 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%  
6352 }%  
6353 \glspostlinkhook  
6354 }
```

\glsxtrlong

```
6355 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\@ns@glsxtrlong}
```

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

```
6356 \newcommand*{\ns@glsxtrlong}[2][\%  
6357 \new@ifnextchar[\@glsxtrlong{#1}{#2}]{\@glsxtrlong{#1}{#2}[]}%  
6358 }
```

Read in the final optional argument:

```
6359 \def\@glsxtrlong#1#2[#3]{%  
6360 \glsdoifexists{#2}%  
6361 {%  
6362 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
6363 \let\glsxtrifwasfirstuse\@secondoftwo  
6364 \let\glsifplural\@secondoftwo  
6365 \let\glsapscase\@firstofthree  
6366 \let\glsinsert\@empty  
6367 \def\glscustomtext{%  
6368 \glsfont{\glsaccesslong{#2}\ifglsxtrininsertinside#3\fi}%  
6369 \ifglsxtrininsertinside\else#3\fi  
6370 }%  
6371 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%  
6372 }
```

```

6372 }%
6373 \glspostlinkhook
6374 }

```

\Glsxtrlong

```

6375 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\@ns@Glsxtrlong}

  Define the un-starred form. Need to determine if there is a final optional argument
6376 \newcommand*{\ns@Glsxtrlong}[2] [] {%
6377   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
6378 }

  Read in the final optional argument:
6379 \def\@Glsxtrlong#1#2[#3]{%
6380   \glsdoifexists{#2}%
6381   {%
6382     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6383     \let\glstrifwasfirstuse\@secondoftwo
6384     \let\glsifplural\@secondoftwo
6385     \let\glscapscase\@thirdofthree
6386     \let\glsinsert\@empty
6387     \def\glscustomtext{%
6388       \glslongfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
6389       \ifglstrinsertinside\else#3\fi
6390     }%
6391     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6392   }%
6393   \glspostlinkhook
6394 }

```

\GLSxtrlong

```

6395 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\@ns@GLSxtrlong}

  Define the un-starred form. Need to determine if there is a final optional argument
6396 \newcommand*{\ns@GLSxtrlong}[2] [] {%
6397   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
6398 }

  Read in the final optional argument:
6399 \def\@GLSxtrlong#1#2[#3]{%
6400   \glsdoifexists{#2}%
6401   {%
6402     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6403     \let\glstrifwasfirstuse\@secondoftwo
6404     \let\glsifplural\@secondoftwo
6405     \let\glscapscase\@thirdofthree
6406     \let\glsinsert\@empty
6407     \def\glscustomtext{%
6408       \mfirstucMakeUppercase
6409       {\glslongfont{\Glsaccesslong{#2}\ifglstrinsertinside#3\fi}%
6410       \ifglstrinsertinside\else#3\fi

```

```

6411     }%
6412     }%
6413     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6414     }%
6415     \glspostlinkhook
6416 }

```

Plural short forms:

\glsxtrshortpl

```

6417 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\@ns@glsxtrshortpl}

```

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

```

6418 \newcommand*{\ns@glsxtrshortpl}[2][ ]{%
6419   \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2}[]}%
6420 }

```

Read in the final optional argument:

```

6421 \def\@glsxtrshortpl#1#2[#3]{%
6422   \glsdoifexists{#2}%
6423   {%
6424     \glssetabbrvfmt{\glscategory{#2}}%
6425     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6426     \let\glsxtrifwasfirstuse\@secondoftwo
6427     \let\glsifplural\@firstoftwo
6428     \let\glscapscase\@firstofthree
6429     \let\glsinsert\@empty
6430     \def\glscustomtext{%
6431       \glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
6432       \ifglsxtrininsertinside\else#3\fi
6433     }%
6434     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6435     }%
6436     \glspostlinkhook
6437 }

```

\Glsxtrshortpl

```

6438 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\@ns@Glsxtrshortpl}

```

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

```

6439 \newcommand*{\ns@Glsxtrshortpl}[2][ ]{%
6440   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2}[]}%
6441 }

```

Read in the final optional argument:

```

6442 \def\@Glsxtrshortpl#1#2[#3]{%
6443   \glsdoifexists{#2}%
6444   {%
6445     \glssetabbrvfmt{\glscategory{#2}}%
6446     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6447     \let\glsxtrifwasfirstuse\@secondoftwo

```

```

6448 \let\glsifplural\@firstoftwo
6449 \let\glscapscase\@secondofthree
6450 \let\glsinsert\@empty
6451 \def\glscustomtext{%
6452     \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
6453     \ifglsxtrinsertinside\else#3\fi
6454 }%
6455 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6456 }%
6457 \glspostlinkhook
6458 }

```

\GLSxtrshortpl

```

6459 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}

    Define the un-starred form. Need to determine if there is a final optional argument
6460 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
6461     \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
6462 }

    Read in the final optional argument:
6463 \def\@GLSxtrshortpl#1#2[#3]{%
6464     \glsdoifexists{#2}%
6465     {%
6466         \glssetabbrvfmt{\glscategory{#2}}%
6467         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6468         \let\glsxtrifwasfirstuse\@secondoftwo
6469         \let\glsifplural\@firstoftwo
6470         \let\glsapspace\@thirdofthree
6471         \let\glsinsert\@empty
6472         \def\glscustomtext{%
6473             \mfirstucMakeUppercase
6474             {\glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
6475             \ifglsxtrinsertinside\else#3\fi
6476         }%
6477     }%
6478     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6479 }%
6480 \glspostlinkhook
6481 }

```

Plural long forms:

\glsxtrlongpl

```

6482 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}

    Define the un-starred form. Need to determine if there is a final optional argument
6483 \newcommand*{\ns@glsxtrlongpl}[2] [] {%
6484     \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2} []}%
6485 }

```


Read in the final optional argument:

```

6486 \def\@glxstrlongpl#1#2[#3]{%
6487   \glstoifexists{#2}%
6488   {%
6489     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6490     \let\glxstrifwasfirstuse\@secondoftwo
6491     \let\gl@ifplural\@firstoftwo
6492     \let\glscapscase\@firstofthree
6493     \let\glinsert\@empty
6494     \def\glscustomtext{%
6495       \glslongfont{\glaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
6496       \ifglxtrininsertinside\else#3\fi
6497     }%
6498     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6499   }%
6500   \glspostlinkhook
6501 }

```

\Glsxtrlongpl

```

6502 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\ns@Glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6503 \newcommand*{\ns@Glsxtrlongpl}[2][]{%
6504   \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2}[]}%
6505 }

```

Read in the final optional argument:

```

6506 \def\@Glsxtrlongpl#1#2[#3]{%
6507   \glstoifexists{#2}%
6508   {%
6509     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6510     \let\glxstrifwasfirstuse\@secondoftwo
6511     \let\gl@ifplural\@firstoftwo
6512     \let\glscapscase\@secondofthree
6513     \let\glinsert\@empty
6514     \def\glscustomtext{%
6515       \glslongfont{\Glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
6516       \ifglxtrininsertinside\else#3\fi
6517     }%
6518     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6519   }%
6520   \glspostlinkhook
6521 }

```

\GLSxtrlongpl

```

6522 \newrobustcmd*{\GLSxtrlongpl}{\@gl@hyp@opt\ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6523 \newcommand*{\ns@GLSxtrlongpl}[2][]{%
6524   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2}[]}%
6525 }

```

Read in the final optional argument:

```
6526 \def\@GLSxtrlongpl#1#2[#3]{%
6527   \glsdoifexists{#2}%
6528   {%
6529     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6530     \let\glxtrifwasfirstuse\@secondoftwo
6531     \let\glsifplural\@firstoftwo
6532     \let\glscapscase\@thirdofthree
6533     \let\glsinsert\@empty
6534     \def\glscustomtext{%
6535       \mfirstucMakeUppercase
6536       {\glslongfont{\glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
6537       \ifglxtrininsertinside\else#3\fi
6538     }%
6539   }%
6540   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6541 }%
6542 \glspostlinkhook
6543 }
```

`\glsetabbrvfmt` Set the current format for the given category (or the abbreviation category if unset).

```
6544 \newcommand*{\glsetabbrvfmt}[1]{%
6545   \ifcsdef{@glsabbrv@current@#1}%
6546   {\glxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
6547   {\glxtr@applyabbrvfmt{@glsabbrv@current@abbreviation}}%
6548 }
```

`glsuseabbrvfont` Provide a way to use the abbreviation font for a given category for arbitrary text.

```
6549 \newrobustcmd*{\glsuseabbrvfont}[2]{\glsetabbrvfmt{#2}\glsabbrvfont{#1}}
```

`\glsuselongfont` Provide a way to use the long font for a given category for arbitrary text.

```
6550 \newrobustcmd*{\glsuselongfont}[2]{\glsetabbrvfmt{#2}\glslongfont{#1}}
```

`sxtrgenabbrvfmt` Similar to `\glsgenacfmt`, but for abbreviations.

```
6551 \newcommand*{\glsxtrgenabbrvfmt}{%
6552   \ifdefempty\glscustomtext
6553   {%
6554     \ifglused\glslabel
6555     {%
```

Subsequent use:

```
6556     \glsifplural
6557     {%
```

Subsequent plural form:

```
6558     \glscapscase
6559     {%
```

Subsequent plural form, don't adjust case:

```
6560      \glxtrsubsequentplfmt{\glslabel}{\glsinsert}%  
6561      }%  
6562      {%
```

Subsequent plural form, make first letter upper case:

```
6563      \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%  
6564      }%  
6565      {%
```

Subsequent plural form, all caps:

```
6566      \mfirstucMakeUppercase  
6567      {\glxtrsubsequentplfmt{\glslabel}{\glsinsert}}%  
6568      }%  
6569      }%  
6570      {%
```

Subsequent singular form

```
6571      \glscapscase  
6572      {%
```

Subsequent singular form, don't adjust case:

```
6573      \glxtrsubsequentfmt{\glslabel}{\glsinsert}%  
6574      }%  
6575      {%
```

Subsequent singular form, make first letter upper case:

```
6576      \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%  
6577      }%  
6578      {%
```

Subsequent singular form, all caps:

```
6579      \mfirstucMakeUppercase  
6580      {\glxtrsubsequentfmt{\glslabel}{\glsinsert}}%  
6581      }%  
6582      }%  
6583      }%  
6584      {%
```

First use:

```
6585      \glsifplural  
6586      {%
```

First use plural form:

```
6587      \glscapscase  
6588      {%
```

First use plural form, don't adjust case:

```
6589      \glxtrfullplformat{\glslabel}{\glsinsert}%  
6590      }%  
6591      {%
```

First use plural form, make first letter upper case:

```
6592      \Glsxtrfullplformat{\glslabel}{\glsinsert}%
6593      }%
6594      {%
```

First use plural form, all caps:

```
6595      \mfirstucMakeUppercase
6596      {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
6597      }%
6598      }%
6599      {%
```

First use singular form

```
6600      \glscapscase
6601      {%
```

First use singular form, don't adjust case:

```
6602      \glsxtrfullformat{\glslabel}{\glsinsert}%
6603      }%
6604      {%
```

First use singular form, make first letter upper case:

```
6605      \Glsxtrfullformat{\glslabel}{\glsinsert}%
6606      }%
6607      {%
```

First use singular form, all caps:

```
6608      \mfirstucMakeUppercase
6609      {\glsxtrfullformat{\glslabel}{\glsinsert}}%
6610      }%
6611      }%
6612      }%
6613      }%
6614      {%
```

User supplied text.

```
6615      \glscustomtext
6616      }%
6617 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
6618 \newcommand*{\glsxtrsubsequentfmt}[2]{%
6619   \glsabbrvfont{\glsaccessshort{#1}\ifglsxtrininsertinside #2\fi}%
6620   \ifglsxtrininsertinside \else#2\fi
6621 }
6622 \let\glsxtrdefaultsubsequentfmt\glsxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural no case change).

```
6623 \newcommand*{\glsxtrsubsequentplfmt}[2]{%
6624   \glsabbrvfont{\glsaccessshortpl{#1}\ifglsxtrininsertinside #2\fi}%
6625   \ifglsxtrininsertinside \else#2\fi
```

```

6626 }
6627 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt

trsubsequentfmt Subsequent use format (singular, first letter uppercase).
6628 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
6629   \glsabbrvfont{\Glsaccessshort{#1}\ifglxtrininsertinside #2\fi}%
6630   \ifglxtrininsertinside \else#2\fi
6631 }
6632 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt

subsequentplfmt Subsequent use format (plural, first letter uppercase).
6633 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
6634   \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrininsertinside #2\fi}%
6635   \ifglxtrininsertinside \else#2\fi
6636 }
6637 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt

```

1.7.1 Abbreviation Styles Setup

```

breviationstyle
6638 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
6639   \ifcsundef{@glsabbrv@dispstyle@setup@#2}%
6640   {%
6641     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
6642   }%
6643   {%
6644     \ifcsstring{@glsabbrv@current@#1}{#2}%
6645     {%
6646       Have abbreviations already been defined for this category?
6647       \ifcsstring{@glsabbrv@current@#1}{#2}%
6648       {%
6649         Style already set.
6650         }%
6651         \def\@glxtr@dostylewarn{%
6652           \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
6653           {%
6654             \def\@glxtr@dostylewarn{\GlossariesWarning{Abbreviation
6655               style has been switched \MessageBreak
6656               for category ‘#1’, \MessageBreak
6657               but there have already been entries \MessageBreak
6658               defined for this category. Unwanted \MessageBreak
6659               side-effects may result}}%
6660             \@endfortrue
6661             }%
6662             \@glxtr@dostylewarn
6663           }%
6664           \csdef{@glsabbrv@current@#1}{#2}%
6665           \glsxtr@applyabbrvstyle{#2}%
6666         }%
6667       }%
6668     }%
6669   }%

```

```

6662     }%
6663 }%
6664 }

```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```

6665 \newcommand*{\glxtr@applyabbrvstyle}[1]{%
6666   \csuse{@glsabbrv@dispstyle@setup@#1}%
6667   \csuse{@glsabbrv@dispstyle@fmts@#1}%
6668 }

```

`\r@applyabbrvfmt` Only apply the style formats.

```

6669 \newcommand*{\glxtr@applyabbrvfmt}[1]{%
6670   \csuse{@glsabbrv@dispstyle@fmts@#1}%
6671 }

```

`\abbreviationstyle` This is different from `\newacronymstyle`. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```

6672 \newcommand*{\newabbreviationstyle}[3]{%
6673   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
6674   {%
6675     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
6676       defined}{}%
6677   }%
6678   {%
6679     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

6680     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
6681     #2}%
6682     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

6683     \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
6684     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
6685     \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
6686     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glxtrsubsequentfmt` etc in case a style changes this.

```

6687     \let\glxtrsubsequentfmt\glxtrdefaultsubsequentfmt
6688     \let\glxtrsubsequentplfmt\glxtrdefaultsubsequentplfmt
6689     \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
6690     \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
6691     #3}%
6692   }%
6693 }

```

`\abbreviationstyle`

```

6694 \newcommand*{\renewabbreviationstyle}[3]{%
6695   \ifcsundef{@glsabbrv@dispstyle@setup@#1}

```

```

6696  {%
6697    \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
6698  }%
6699  {%
6700    \csdef{@glsabbrv@dispstyle@setup@#1}{%
      Initialise hook to do nothing. The style may change this.
6701      \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
6702      #2}%
6703    \csdef{@glsabbrv@dispstyle@fmts@#1}{%
      Assume in-line form is the same as first use. The style may change this.
6704      \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
6705      \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
6706      \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
6707      \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
6708      #3}%
6709    }%
6710 }

```

abbreviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```

6711 \newcommand*{\letabbreviationstyle}[2]{%
6712   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
6713   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
6714 }

```

deprecated@abbrstyle `\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}`

Define a synonym for a deprecated abbreviation style.

```

6715 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
6716   \csdef{@glsabbrv@dispstyle@setup@#1}{%
6717     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
6718     \csuse{@glsabbrv@dispstyle@setup@#2}%
6719   }%
6720   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
6721 }

```

deprecatedAbbrStyle Generate warning for deprecated style use.

```

6722 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
6723   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
6724     use ‘#2’ instead}%
6725 }

```

eAbbrStyleSetup

```

6726 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
6727   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%

```

```

6728 {%
6729     \PackageError{glossaries-extra}%
6730     {Unknown abbreviation style definitions ‘#1’}{}%
6731 }%
6732 {%
6733     \csname @glsabbrv@dispstyle@setup@#1\endcsname
6734 }%
6735 }

```

seAbbrStyleFmts

```

6736 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
6737     \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
6738     {%
6739         \PackageError{glossaries-extra}%
6740         {Unknown abbreviation style formats ‘#1’}{}%
6741     }%
6742     {%
6743         \csname @glsabbrv@dispstyle@fmts@#1\endcsname
6744     }%
6745 }

```

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.

```

6746 \newif\ifglsxtrinsertinside
6747 \glsxtrinsertinsidefalse

```

trlongshortname

```

6748 \newcommand*{\glsxtrlongshortname}{%
6749     \protect\glsabbrvfont{\the\glsshorttok}%
6750 }

```

long-short

```

6751 \newabbreviationstyle{long-short}%
6752 {%
6753     \renewcommand*{\CustomAbbreviationFields}{%
6754         name={\glsxtrlongshortname},
6755         sort={\the\glsshorttok},

```



```

6756 first={\protect\glsfirstlongfont{\the\glslongtok}%
6757 \protect\glsxtrfullsep{\the\glslabeltok}%
6758 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
6759 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
6760 \protect\glsxtrfullsep{\the\glslabeltok}%
6761 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

6762 plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
6763 description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

6764 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6765 \glsasattribute{\the\glslabeltok}{regular}%
6766 {%
6767 \glssetattribute{\the\glslabeltok}{regular}{false}%
6768 }%
6769 {}%
6770 }%
6771 }%
6772 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6773 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
6774 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
6775 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
6776 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
6777 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

6778 \renewcommand*{\glsxtrfullformat}[2]{%
6779 \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
6780 \ifglsxtrininsertinside\else##2\fi
6781 \glsxtrfullsep{##1}%
6782 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
6783 }%
6784 \renewcommand*{\glsxtrfullplformat}[2]{%
6785 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
6786 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
6787 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
6788 }%
6789 \renewcommand*{\Glsxtrfullformat}[2]{%
6790 \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
6791 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
6792 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
6793 }%
6794 \renewcommand*{\Glsxtrfullplformat}[2]{%
6795 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
6796 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
6797 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
6798 }%
6799 }

```

Set this as the default style for general abbreviations:

```
6800 \setabbreviationstyle{long-short}
```

ngshortdescsort

```
6801 \newcommand*{\glxtrlongshortdescsort}{%
6802 \expandonce\glxtrorglong\space (\expandonce\glxtrorgshort)%
6803 }
```

ngshortdescname

```
6804 \newcommand*{\glxtrlongshortdescname}{%
6805 \protect\glslongfont{\the\glslongtok}
6806 \glxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}}%
6807 }
```

long-short-desc User supplies description. The long form is included in the name.

```
6808 \newabbreviationstyle{long-short-desc}%
6809 {%
6810 \renewcommand*{\CustomAbbreviationFields}{%
6811 name={\glxtrlongshortdescname},
6812 sort={\glxtrlongshortdescsort},%
6813 first={\protect\glsfirstlongfont{\the\glslongtok}%
6814 \protect\glxtrfullsep{\the\glslabeltok}%
6815 \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
6816 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
6817 \protect\glxtrfullsep{\the\glslabeltok}%
6818 \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
```

The text key should only have the short form.

```
6819 text={\protect\glsabbrvfont{\the\glsshorttok}},%
6820 plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
6821 }%
```

Unset the regular attribute if it has been set.

```
6822 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6823 \glshasattribute{\the\glslabeltok}{regular}%
6824 {%
6825 \glssetattribute{\the\glslabeltok}{regular}{false}%
6826 }%
6827 {}%
6828 }%
6829 }%
6830 {%
6831 \GlsXtrUseAbbrStyleFmts{long-short}%
6832 }
```

trshortlongname

```
6833 \newcommand*{\glxtrshortlongname}{%
6834 \protect\glsabbrvfont{\the\glsshorttok}%
6835 }
```

short-long Short form followed by long form in parenthesis on first use.

```
6836 \newabbreviationstyle{short-long}%
6837 {%
6838   \renewcommand*{\CustomAbbreviationFields}{%
6839     name={\glxtrshortlongname},
6840     sort={\the\glsshorttok},
6841     description={\the\gslongtok},%
6842     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
6843       \protect\glxtrfullsep{\the\glslabeltok}%
6844       \glxtrparen{\protect\glsfirstlongfont{\the\gslongtok}}},%
6845     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
6846       \protect\glxtrfullsep{\the\glslabeltok}%
6847       \glxtrparen{\protect\glsfirstlongfont{\the\gslongpltok}}},%
6848     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
6849 }
```

Unset the regular attribute if it has been set.

```
6849 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6850   \glshasattribute{\the\glslabeltok}{regular}%
6851   {%
6852     \glissetattribute{\the\glslabeltok}{regular}{false}%
6853   }%
6854   {}%
6855 }%
6856 }%
6857 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
6858 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
6859 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
6860 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
6861 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
6862 \renewcommand*{\gslongfont}[1]{\gslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
6863 \renewcommand*{\glxtrfullformat}[2]{%
6864   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6865   \ifglxtrininsertinside\else##2\fi
6866   \glxtrfullsep{##1}%
6867   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
6868 }%
6869 \renewcommand*{\glxtrfullplformat}[2]{%
6870   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
6871   \ifglxtrininsertinside\else##2\fi
6872   \glxtrfullsep{##1}%
6873   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
6874 }%
6875 \renewcommand*{\Glsxtrfullformat}[2]{%
6876   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6877   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
6878 }
```

```

6878 \glstrparen{\glfirstlongfont{\glaccesslong{##1}}}%
6879 }%
6880 \renewcommand*{\Glsxtrfullplformat}[2]{%
6881 \glfirstabbrvfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
6882 \ifglstrinsertinside\else##2\fi\glxtrfullsep{##1}%
6883 \glstrparen{\glfirstlongfont{\glaccesslongpl{##1}}}%
6884 }%
6885 }

```

ortlongdescsort

```

6886 \newcommand*{\glstrshortlongdescsort}{\the\glsshorttok}

```

ortlongdescname

```

6887 \newcommand*{\glstrshortlongdescname}{%
6888 \protect\glsabbrvfont{\the\glsshorttok}
6889 \glstrparen{\protect\glslongfont{\the\glslongtok}}}%
6890 }

```

short-long-desc User supplies description. The long form is included in the name.

```

6891 \newabbreviationstyle{short-long-desc}%
6892 {%
6893 \renewcommand*{\CustomAbbreviationFields}{%
6894 name={\glstrshortlongdescname},
6895 sort={\glstrshortlongdescsort},
6896 first={\protect\glfirstabbrvfont{\the\glsshorttok}}%
6897 \protect\glxtrfullsep{\the\glslabeltok}}%
6898 \glstrparen{\protect\glfirstlongfont{\the\glslongtok}}},%
6899 firstplural={\protect\glfirstabbrvfont{\the\glsshortpltok}}%
6900 \protect\glxtrfullsep{\the\glslabeltok}}%
6901 \glstrparen{\protect\glfirstlongfont{\the\glslongpltok}}},%
6902 text={\protect\glsabbrvfont{\the\glsshorttok}},%
6903 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
6904 }%

```

Unset the regular attribute if it has been set.

```

6905 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6906 \glshasattribute{\the\glslabeltok}{regular}%
6907 {%
6908 \glissetattribute{\the\glslabeltok}{regular}{false}%
6909 }%
6910 {}%
6911 }%
6912 }%
6913 {%
6914 \GlsXtrUseAbbrStyleFmts{short-long}%
6915 }

```

ongfootnotefont Only used by the “footnote” styles.

```
6916 \newcommand*{\glfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

ongfootnotefont Only used by the “footnote” styles.

```
6917 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%
```

xtrabbrvfootnote `\glxtrabbrvfootnote{<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`).

```
6918 \newcommand*{\glxtrabbrvfootnote}[2]{\footnote{#2}}
```

xtrfootnotename

```
6919 \newcommand*{\glxtrfootnotename}{%  
6920   \protect\glsabbrvfont{\the\glsshorttok}%  
6921 }
```

footnote Short form followed by long form in footnote on first use.

```
6922 \newabbreviationstyle{footnote}%  
6923 {%  
6924   \renewcommand*{\CustomAbbreviationFields}{%  
6925     name={\glxtrfootnotename},  
6926     sort={\the\glsshorttok},  
6927     description={\the\glslongtok},%  
  
6928     first={\protect\glfirstabbrvfont{\the\glsshorttok}%  
6929       \protect\glxtrabbrvfootnote{\the\glslabeltok}%  
6930       {\protect\glfirstlongfootnotefont{\the\glslongtok}}},%  
6931     firstplural={\protect\glfirstabbrvfont{\the\glsshortpltok}%  
6932       \protect\glxtrabbrvfootnote{\the\glslabeltok}%  
6933       {\protect\glfirstlongfootnotefont{\the\glslongpltok}}},%  
  
6934     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.

```
6935 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
6936   \glsssetAttribute{\the\glslabeltok}{nohyperfirst}{true}%  
6937   \glshasattribute{\the\glslabeltok}{regular}%  
6938   {%  
6939     \glsssetAttribute{\the\glslabeltok}{regular}{false}%  
6940   }%  
6941   {}%  
6942 }
```

6943 }%
 6944 {%

In case the user wants to mix and match font styles, these are redefined here.

```
6945 \renewcommand*\abbrvpluralsuffix{\glstrabbrvpluralsuffix}%
6946 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
6947 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
6948 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
6949 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
6950 \renewcommand*\glstrfullformat[2]{%
6951   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
6952   \ifglstrinsertinside\else##2\fi
6953   \protect\glstrabbrvfootnote{##1}%
6954   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
6955 }%
6956 \renewcommand*\glstrfullplformat[2]{%
6957   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
6958   \ifglstrinsertinside\else##2\fi
6959   \protect\glstrabbrvfootnote{##1}%
6960   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
6961 }%
6962 \renewcommand*\Glsxtrfullformat[2]{%
6963   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
6964   \ifglstrinsertinside\else##2\fi
6965   \protect\glstrabbrvfootnote{##1}%
6966   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
6967 }%
6968 \renewcommand*\Glsxtrfullplformat[2]{%
6969   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
6970   \ifglstrinsertinside\else##2\fi
6971   \protect\glstrabbrvfootnote{##1}%
6972   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
6973 }%
```

The first use full form and the inline full form use the short (long) style.

```
6974 \renewcommand*\glstrinlinefullformat[2]{%
6975   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
6976   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
6977   \glstrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
6978 }%
6979 \renewcommand*\glstrinlinefullplformat[2]{%
6980   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
6981   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
6982   \glstrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
6983 }%
6984 \renewcommand*\Glsxtrinlinefullformat[2]{%
6985   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
6986   \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
6987   \glstrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
6988 }
```

```

6988 }%
6989 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
6990   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6991   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6992   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
6993 }%
6994 }

```

short-footnote

```

6995 \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.

```

6996 \newabbreviationstyle{postfootnote}%
6997 {%
6998   \renewcommand*{\CustomAbbreviationFields}{%
6999     name={\glxtrfootnotename},
7000     sort={\the\glsshorttok},
7001     description={\the\glslongtok},%
7002     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7003     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7004     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.

```

7005   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7006     \csdef{glxtrpostlink\glscategorylabel}{%
7007       \glxtrifwasfirstuse
7008       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7009         \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7010         {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
7011       }%
7012     }%
7013   }%
7014   \glshasattribute{\the\glslabeltok}{regular}%
7015   {%
7016     \glissetattribute{\the\glslabeltok}{regular}{false}%
7017   }%
7018   {}%
7019 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

7020 \renewcommand*{\glxtrsetupfulldefs}{%
7021   \let\glxtrifwasfirstuse\@secondoftwo

```

```

7022 }%
7023 }%
7024 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7025 \renewcommand*\abbrvpluralsuffix{\glxtrabbrvpluralsuffix}%
7026 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7027 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7028 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
7029 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

7030 \renewcommand*\glxtrfullformat[2]{%
7031   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7032   \ifglxtrininsertinside\else##2\fi
7033 }%
7034 \renewcommand*\glxtrfullplformat[2]{%
7035   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7036   \ifglxtrininsertinside\else##2\fi
7037 }%
7038 \renewcommand*\Glsxtrfullformat[2]{%
7039   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7040   \ifglxtrininsertinside\else##2\fi
7041 }%
7042 \renewcommand*\Glsxtrfullplformat[2]{%
7043   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7044   \ifglxtrininsertinside\else##2\fi
7045 }%

```

The first use full form and the inline full form use the short (long) style.

```

7046 \renewcommand*\glxtrininlinefullformat[2]{%
7047   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7048   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7049   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7050 }%
7051 \renewcommand*\glxtrininlinefullplformat[2]{%
7052   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7053   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7054   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7055 }%
7056 \renewcommand*\Glsxtrininlinefullformat[2]{%
7057   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7058   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7059   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7060 }%
7061 \renewcommand*\Glsxtrininlinefullplformat[2]{%
7062   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7063   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7064   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7065 }%
7066 }

```


rt-postfootnote

```
7067 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

shortnolongname

```
7068 \newcommand*{\glxtrshortnolongname}{%
7069   \protect\glsabbrvfont{\the\glsshorttok}%
7070 }
```

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.

```
7071 \newabbreviationstyle{short}%
7072 {%
7073   \renewcommand*{\CustomAbbreviationFields}{%
7074     name={\glxtrshortnolongname},
7075     sort={\the\glsshorttok},
7076     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7077     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7078     text={\protect\glsabbrvfont{\the\glsshorttok}},
7079     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7080     description={\the\glslongtok}}%
7081   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7082     \glssetattribute{\the\glslabeltok}{regular}{true}}%
7083 }%
7084 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7085 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7086 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7087 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7088 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7089 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
7090 \renewcommand*{\glxtrinlinefullformat}[2]{%
7091   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
7092   \ifglxtrininsertinside##2\fi}%
7093   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7094   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7095 }%
7096 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7097   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
7098   \ifglxtrininsertinside##2\fi}%
7099   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7100   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7101 }%
7102 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7103   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
```

```

7104     \ifglxtrinsertinside##2\fi}%
7105     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7106     \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7107 }%
7108 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7109     \protect\glsfirstabbrvfont{\Glsaccessshortpl{##1}%
7110         \ifglxtrinsertinside##2\fi}%
7111     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7112     \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7113 }%

```

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.

```

7114 \renewcommand*{\glxtrfullformat}[2]{%
7115     \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7116     \ifglxtrinsertinside\else##2\fi
7117 }%
7118 \renewcommand*{\glxtrfullplformat}[2]{%
7119     \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7120     \ifglxtrinsertinside\else##2\fi
7121 }%
7122 \renewcommand*{\Glsxtrfullformat}[2]{%
7123     \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7124     \ifglxtrinsertinside\else##2\fi
7125 }%
7126 \renewcommand*{\Glsxtrfullplformat}[2]{%
7127     \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7128     \ifglxtrinsertinside\else##2\fi
7129 }%
7130 }

```

Set this as the default style for acronyms:

```

7131 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

7132 \letabbreviationstyle{short-nolong}{short}

```

short-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

7133 \newabbreviationstyle{short-nolong-noreg}%
7134 {%
7135     \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

7136 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7137     \glshasattribute{\the\glslabeltok}{regular}%
7138     {%
7139         \glissetattribute{\the\glslabeltok}{regular}{false}%
7140     }%
7141     {}%
7142 }%

```

```

7143 }%
7144 {%
7145   \GlsXtrUseAbbrStyleFmts{short-nolong}%
7146 }

```

trshortdescname

```

7147 \newcommand*{\glxtrshortdescname}{%
7148   \protect\glsabbrvfont{\the\glsshorttok}%
7149 }

```

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.

```

7150 \newabbreviationstyle{short-desc}%
7151 {%
7152   \renewcommand*{\CustomAbbreviationFields}{%
7153     name={\glxtrshortdescname},
7154     sort={\the\glsshorttok},
7155     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7156     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7157     text={\protect\glsabbrvfont{\the\glsshorttok}},
7158     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7159     description={\the\glslongtok}}%
7160   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7161     \glssetattribute{\the\glslabeltok}{regular}{true}}%
7162 }%
7163 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7164 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7165 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7166 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7167 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7168 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7169 \renewcommand*{\glxtrinlinefullformat}[2]{%
7170   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7171   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7172   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7173 }%
7174 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7175   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7176   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7177   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7178 }%
7179 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7180   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7181   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7182   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7183 }%

```

```

7184 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7185   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7186   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7187   \glsxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7188 }%

```

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.

```

7189 \renewcommand*{\glsxtrfullformat}[2]{%
7190   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7191   \ifglxtrinsertinside\else##2\fi
7192 }%
7193 \renewcommand*{\glsxtrfullplformat}[2]{%
7194   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7195   \ifglxtrinsertinside\else##2\fi
7196 }%
7197 \renewcommand*{\Glsxtrfullformat}[2]{%
7198   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7199   \ifglxtrinsertinside\else##2\fi
7200 }%
7201 \renewcommand*{\Glsxtrfullplformat}[2]{%
7202   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7203   \ifglxtrinsertinside\else##2\fi
7204 }%
7205 }

```

ort-nolong-desc

```

7206 \letabbreviationstyle{short-nolong-desc}{short-desc}

```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

7207 \newabbreviationstyle{short-nolong-desc-noreg}%
7208 {%
7209   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

```

Unset the regular attribute if it has been set.

```

7210 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7211   \glshasattribute{\the\glslabeltok}{regular}%
7212   {%
7213     \glssetattribute{\the\glslabeltok}{regular}{false}%
7214   }%
7215   {%
7216   }%
7217 }%
7218 {%
7219   \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
7220 }

```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

7221 \newabbreviationstyle{nolong-short}%
7222 {%
7223   \GlsXtrUseAbbrStyleSetup{short-nolong}%
7224 }%
7225 {%
7226   \GlsXtrUseAbbrStyleFmts{short-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

7227 \renewcommand*{\glxtrinlinefullformat}[2]{%
7228   \protect\glsfirstlongfont{\glsaccesslong{##1}}%
7229   \ifglxtrininsertinside##2\fi}%
7230 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7231 \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7232 }%
7233 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7234   \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
7235   \ifglxtrininsertinside##2\fi}%
7236 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7237 \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7238 }%
7239 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7240   \protect\glsfirstlongfont{\glsaccesslong{##1}}%
7241   \ifglxtrininsertinside##2\fi}%
7242 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7243 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
7244 }%
7245 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7246   \protect\glsfirstlongfont{\glsaccesslongpl{##1}}%
7247   \ifglxtrininsertinside##2\fi}%
7248 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7249 \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
7250 }%
7251 }

```

nolong-short-noreg Like **nolong-short** but doesn't set the regular attribute.

```

7252 \newabbreviationstyle{nolong-short-noreg}%
7253 {%
7254   \GlsXtrUseAbbrStyleSetup{nolong-short}%

```

Unset the regular attribute if it has been set.

```

7255 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7256   \glshasattribute{\the\glslabeltok}{regular}%
7257   {%
7258     \glissetattribute{\the\glslabeltok}{regular}{false}%
7259   }%
7260   {}%
7261 }%
7262 }%
7263 {%
7264   \GlsXtrUseAbbrStyleFmts{nolong-short}%

```

7265 }

noshortdescname

```
7266 \newcommand*{\glxtrlongnoshortdescname}{%
7267   \protect\glslongfont{\the\glslongtok}%
7268 }
```

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.

```
7269 \newabbreviationstyle{long-desc}%
7270 {%
7271   \renewcommand*{\CustomAbbreviationFields}{%
7272     name={\glxtrlongnoshortdescname},
7273     sort={\the\glslongtok},
7274     first={\protect\glsfirstlongfont{\the\glslongtok}},
7275     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7276     text={\glslongfont{\the\glslongtok}},
7277     plural={\glslongfont{\the\glslongpltok}}}%
7278   }%
7279   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7280     \glssetattribute{\the\glslabeltok}{regular}{true}}%
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 format for subsequent use (not used when the regular attribute is set).

```
7288   \renewcommand*{\glxtrsubsequentfmt}[2]{%
7289     \glslongfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7290     \ifglxtrinsertinside \else##2\fi
7291   }%
7292   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
7293     \glslongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7294     \ifglxtrinsertinside \else##2\fi
7295   }%
7296   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
7297     \glslongfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7298     \ifglxtrinsertinside \else##2\fi
7299   }%
7300   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
7301     \glslongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7302     \ifglxtrinsertinside \else##2\fi
7303   }%
```

The inline full form displays the long format followed by the short form in parentheses.

```

7304 \renewcommand*{\glxtrinlinefullformat}[2]{%
7305   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7306   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7307   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7308 }%
7309 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7310   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7311   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7312   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7313 }%
7314 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7315   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7316   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7317   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7318 }%
7319 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7320   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7321   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7322   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7323 }%

```

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.

```

7324 \renewcommand*{\glxtrfullformat}[2]{%
7325   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7326   \ifglxtrinsertinside\else##2\fi
7327 }%
7328 \renewcommand*{\glxtrfullplformat}[2]{%
7329   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7330   \ifglxtrinsertinside\else##2\fi
7331 }%
7332 \renewcommand*{\Glsxtrfullformat}[2]{%
7333   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7334   \ifglxtrinsertinside\else##2\fi
7335 }%
7336 \renewcommand*{\Glsxtrfullplformat}[2]{%
7337   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7338   \ifglxtrinsertinside\else##2\fi
7339 }%
7340 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```
7341 \letabbreviationstyle{long-noshort-desc}{long-desc}
```

hort-desc-noreg Like long-noshort-desc but doesn't set the regular attribute.

```

7342 \newabbreviationstyle{long-noshort-desc-noreg}%
7343 {%
7344   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%

```

Unset the regular attribute if it has been set.

```
7345 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7346   \glshasattribute{\the\glslabeltok}{regular}%
7347   {%
7348     \glsselattribute{\the\glslabeltok}{regular}{false}%
7349   }%
7350   {}%
7351 }%
7352}%
7353{%
7354   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
7355 }
```

longnoshortname

```
7356 \newcommand*{\glsxtrlongnoshortname}{%
7357   \protect\glsabbrvfont{\the\glsshorttok}%
7358 }
```

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.

```
7359 \newabbreviationstyle{long}%
7360 {%
7361   \renewcommand*{\CustomAbbreviationFields}{%
7362     name={\glsxtrlongnoshortname},
7363     sort={\the\glsshorttok},
7364     first={\protect\glsfirstlongfont{\the\glslongtok}},
7365     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7366     text={\glslongfont{\the\glslongtok}},
7367     plural={\glslongfont{\the\glslongpltok}},%
7368     description={\the\glslongtok}%
7369   }%
7370   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7371     \glsselattribute{\the\glslabeltok}{regular}{true}}%
7372}%
7373{%
7374   \GlsXtrUseAbbrStyleFmts{long-desc}%
7375 }
```

long-noshort Provide a synonym that matches similar styles.

```
7376 \letabbreviationstyle{long-noshort}{long}
```

g-noshort-noreg Like long-noshort but doesn't set the regular attribute.

```
7377 \newabbreviationstyle{long-noshort-noreg}%
7378 {%
7379   \GlsXtrUseAbbrStyleSetup{long-noshort}%
7380 }
```

Unset the regular attribute if it has been set.

```
7380 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7381   \glsselattribute{\the\glslabeltok}{regular}{false}%
7382 }
```



```

7381 \glshasattribute{\the\glslabeltok}{regular}%
7382 {%
7383 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
7384 }%
7385 {}%
7386 }%
7387 }%
7388 {%
7389 \GlsXtrUseAbbrStyleFmts{long-noshort}%
7390 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glsxtrscfont` Maintained for backward-compatibility.

```
7391 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
7392 \newcommand*{\glsabbrvscfont}{\glsxtrscfont}
```

`\glsxtrfirstscfont` Maintained for backward-compatibility.

```
7393 \newcommand*{\glsxtrfirstscfont}[1]{\glsabbrvscfont{#1}}
```

`\glstfirstabbrvscfont` Added for consistent naming.

```
7394 \newcommand*{\glstfirstabbrvscfont}{\glsxtrfirstscfont}
```

and for the default short form suffix:

`\glsxtrscsuffix`

```
7395 \newcommand*{\glsxtrscsuffix}{\glstextup{\glxtrabbrvpluralsuffix}}
```

`long-short-sc`

```

7396 \newabbreviationstyle{long-short-sc}%
7397 {%
7398 \renewcommand*{\CustomAbbreviationFields}{%
7399 name={\glsxtrlongshortname},
7400 sort={\the\glsshorttok},
7401 first={\protect\glstfirstlongdefaultfont{\the\glslongtok}%
7402 \protect\glsxtrfullsep{\the\glslabeltok}%
7403 \glsxtrparen{\protect\glstfirstabbrvscfont{\the\glsshorttok}}},%
7404 firstplural={\protect\glstfirstlongdefaultfont{\the\glslongpltok}%
7405 \protect\glsxtrfullsep{\the\glslabeltok}%
7406 \glsxtrparen{\protect\glstfirstabbrvscfont{\the\glsshortpltok}}},%
7407 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}},%
7408 description={\the\glslongtok}}%
7409 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7410 \glshasattribute{\the\glslabeltok}{regular}%
7411 {%

```

```

7412 \glsetattribute{\the\glslabeltok}{regular}{false}%
7413 }%
7414 {}%
7415 }%
7416 }%
7417 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7418 \renewcommand*\abbrvpluralsuffix{\protect\glxtrscsuffix}%
7419 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
7420 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%

```

Use the default long fonts.

```

7421 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7422 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7423 \renewcommand*\glsxtrfullformat[2]{%
7424 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7425 \ifglsxtrininsertinside\else##2\fi
7426 \glsxtrfullsep{##1}%
7427 \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7428 }%
7429 \renewcommand*\glsxtrfullplformat[2]{%
7430 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7431 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7432 \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7433 }%
7434 \renewcommand*\Glsxtrfullformat[2]{%
7435 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7436 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7437 \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7438 }%
7439 \renewcommand*\Glsxtrfullplformat[2]{%
7440 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7441 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7442 \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7443 }%
7444 }

```

g-short-sc-desc

```

7445 \newabbreviationstyle{long-short-sc-desc}%
7446 {%
7447 \renewcommand*\CustomAbbreviationFields{%
7448 name={\glsxtrlongshortdescname},
7449 sort={\glsxtrlongshortdescsort},%
7450 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7451 \protect\glsxtrfullsep{\the\glslabeltok}%
7452 \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7453 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%

```

```

7454 \protect\glxtrfullsep{\the\glslabeltok}%
7455 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7456 text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7457 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7458 }%

```

Unset the regular attribute if it has been set.

```

7459 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7460 \glshasattribute{\the\glslabeltok}{regular}%
7461 {%
7462 \glissetattribute{\the\glslabeltok}{regular}{false}%
7463 }%
7464 {}%
7465 }%
7466 }%
7467 {%

```

As long-short-sc style:

```

7468 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
7469 }

```

Now the short (long) version

```

7470 \newabbreviationstyle{short-sc-long}%
7471 {%
7472 \renewcommand*{\CustomAbbreviationFields}{%
7473 name={\glxtrshortlongname},
7474 sort={\the\glsshorttok},
7475 description={\the\gslongtok},%
7476 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}}%
7477 \protect\glxtrfullsep{\the\glslabeltok}%
7478 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongtok}}},%
7479 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}}%
7480 \protect\glxtrfullsep{\the\glslabeltok}%
7481 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongpltok}}},%
7482 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

7483 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7484 \glshasattribute{\the\glslabeltok}{regular}%
7485 {%
7486 \glissetattribute{\the\glslabeltok}{regular}{false}%
7487 }%
7488 {}%
7489 }%
7490 }%
7491 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7492 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7493 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7494 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%

```

```

7495 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7496 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7497 \renewcommand*{\glsxtrfullformat}[2]{%
7498   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7499   \ifglsxtrininsertinside\else##2\fi
7500   \glsxtrfullsep{##1}%
7501   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7502 }%
7503 \renewcommand*{\glsxtrfullplformat}[2]{%
7504   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7505   \ifglsxtrininsertinside\else##2\fi
7506   \glsxtrfullsep{##1}%
7507   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7508 }%
7509 \renewcommand*{\Glsxtrfullformat}[2]{%
7510   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7511   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7512   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7513 }%
7514 \renewcommand*{\Glsxtrfullplformat}[2]{%
7515   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7516   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7517   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7518 }%
7519 }

```

As before but user provides description

```

7520 \newabbreviationstyle{short-sc-long-desc}%
7521 {%
7522   \renewcommand*{\CustomAbbreviationFields}{%
7523     name={\glsxtrshortlongdescname},
7524     sort={\glsxtrshortlongdescsort},
7525     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
7526       \protect\glsxtrfullsep{\the\glslabeltok}%
7527       \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
7528     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7529       \protect\glsxtrfullsep{\the\glslabeltok}%
7530       \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
7531     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7532     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
7533   }%

```

Unset the regular attribute if it has been set.

```

7534 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7535   \glshasattribute{\the\glslabeltok}{regular}%
7536   {%
7537     \glssetattribute{\the\glslabeltok}{regular}{false}%
7538   }%

```

```

7539    {}%
7540  }%
7541 }%
7542 {%
    As short-sc-long style:
7543   \GlsXtrUseAbbrStyleFmts{short-sc-long}%
7544 }

```

short-sc

```

7545 \newabbreviationstyle{short-sc}%
7546 {%
7547   \renewcommand*{\CustomAbbreviationFields}{%
7548     name={\glstrshortnolongname},
7549     sort={\the\glsshorttok},
7550     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
7551     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
7552     text={\protect\glsabbrvscfont{\the\glsshorttok}},
7553     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
7554     description={\the\glslongtok}}%
7555   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7556     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
7557 }%
7558 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7559   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrscsuffix}%
7560   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7561   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
7562   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7563   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7564   \renewcommand*{\glxtrinlinefullformat}[2]{%
7565     \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
7566     \ifglxtrininsertinside##2\fi}%
7567   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7568   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7569 }%
7570   \renewcommand*{\glxtrinlinefullplformat}[2]{%
7571     \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
7572     \ifglxtrininsertinside##2\fi}%
7573   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7574   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7575 }%

7576   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7577     \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}}%
7578     \ifglxtrininsertinside##2\fi}%
7579   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7580   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%

```

```

7581 }%
7582 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7583   \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}%
7584     \ifglxtrinsertinside##2\fi}%
7585   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7586   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
7587 }%

```

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.

```

7588 \renewcommand*{\glxtrfullformat}[2]{%
7589   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7590   \ifglxtrinsertinside\else##2\fi
7591 }%
7592 \renewcommand*{\glxtrfullplformat}[2]{%
7593   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7594   \ifglxtrinsertinside\else##2\fi
7595 }%
7596 \renewcommand*{\Glsxtrfullformat}[2]{%
7597   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7598   \ifglxtrinsertinside\else##2\fi
7599 }%
7600 \renewcommand*{\Glsxtrfullplformat}[2]{%
7601   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7602   \ifglxtrinsertinside\else##2\fi
7603 }%
7604 }

```

short-sc-nolong

```

7605 \letabbreviationstyle{short-sc-nolong}{short-sc}

```

short-sc-desc

```

7606 \newabbreviationstyle{short-sc-desc}%
7607 {%
7608   \renewcommand*{\CustomAbbreviationFields}{%
7609     name={\glxtrshortdescname},
7610     sort={\the\glsshorttok},
7611     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
7612     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
7613     text={\protect\glsabbrvscfont{\the\glsshorttok}},
7614     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
7615     description={\the\glslongtok}}%
7616 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7617   \glssetattribute{\the\glslabeltok}{regular}{true}}%
7618 }%
7619 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7620 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7621 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%

```

```

7622 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
7623 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7624 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7625 \renewcommand*\glsxtrinlinefullformat}[2]{%
7626   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7627   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7628   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7629 }%
7630 \renewcommand*\glsxtrinlinefullplformat}[2]{%
7631   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7632   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7633   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7634 }%
7635 \renewcommand*\Glsxtrinlinefullformat}[2]{%
7636   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7637   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7638   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
7639 }%
7640 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
7641   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7642   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7643   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
7644 }%

```

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.

```

7645 \renewcommand*\glsxtrfullformat}[2]{%
7646   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7647   \ifglsxtrininsertinside\else##2\fi
7648 }%
7649 \renewcommand*\glsxtrfullplformat}[2]{%
7650   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7651   \ifglsxtrininsertinside\else##2\fi
7652 }%
7653 \renewcommand*\Glsxtrfullformat}[2]{%
7654   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7655   \ifglsxtrininsertinside\else##2\fi
7656 }%
7657 \renewcommand*\Glsxtrfullplformat}[2]{%
7658   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7659   \ifglsxtrininsertinside\else##2\fi
7660 }%
7661 }

```

-sc-nolong-desc

```

7662 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}

```

nolong-short-sc

```

7663 \newabbreviationstyle{nolong-short-sc}%
7664 {%
7665   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
7666 }%
7667 {%
7668   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

7669 \renewcommand*{\glxtrinlinefullformat}[2]{%
7670   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}}%
7671   \ifglxtrinsertinside##2\fi}%
7672 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7673 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7674 }%
7675 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7676   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}%
7677   \ifglxtrinsertinside##2\fi}%
7678 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7679 \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7680 }%
7681 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7682   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}}%
7683   \ifglxtrinsertinside##2\fi}%
7684 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7685 \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
7686 }%
7687 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7688   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}%
7689   \ifglxtrinsertinside##2\fi}%
7690 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7691 \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
7692 }%
7693 }

```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like \glxtrshort.

```

7694 \newabbreviationstyle{long-noshort-sc}%
7695 {%
7696   \renewcommand*{\CustomAbbreviationFields}{%
7697     name={\glxtrlongnoshortname},
7698     sort={\the\glsshorttok},
7699     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
7700     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
7701     text={\protect\glslongdefaultfont{\the\glslongtok}},
7702     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
7703     description={\the\glslongtok}%
7704   }%
7705   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7706     \glssetattribute{\the\glslabeltok}{regular}{true}}%
7707 }%

```


7708 {%

Use smallcaps and adjust the plural suffix to revert to upright.

```
7709 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7710 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
7711 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvscfont{##1}}%
7712 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
7713 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
7714 \renewcommand*{\glxtrsubsequentfmt}[2]{%
7715   \glsslongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
7716   \ifglxtrininsertinside \else##2\fi
7717 }%
7718 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
7719   \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
7720   \ifglxtrininsertinside \else##2\fi
7721 }%
7722 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
7723   \glsslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
7724   \ifglxtrininsertinside \else##2\fi
7725 }%
7726 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
7727   \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
7728   \ifglxtrininsertinside \else##2\fi
7729 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
7730 \renewcommand*{\glxtrinlinefullformat}[2]{%
7731   \glssfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7732   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7733   \glxtrparen{\protect\glssfirstabbrvscfont{\glssaccessshort{##1}}}%
7734 }%
7735 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7736   \glssfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7737   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7738   \glxtrparen{\protect\glssfirstabbrvscfont{\glssaccessshortpl{##1}}}%
7739 }%
7740 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7741   \glssfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7742   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7743   \glxtrparen{\protect\glssfirstabbrvscfont{\glssaccessshort{##1}}}%
7744 }%
7745 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7746   \glssfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7747   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7748   \glxtrparen{\protect\glssfirstabbrvscfont{\glssaccessshortpl{##1}}}%
7749 }%
```

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.

```

7750 \renewcommand*{\glxtrfullformat}[2]{%
7751   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7752   \ifglxtrinsertinside\else##2\fi
7753 }%
7754 \renewcommand*{\glxtrfullplformat}[2]{%
7755   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7756   \ifglxtrinsertinside\else##2\fi
7757 }%
7758 \renewcommand*{\Glsxtrfullformat}[2]{%
7759   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7760   \ifglxtrinsertinside\else##2\fi
7761 }%
7762 \renewcommand*{\Glsxtrfullplformat}[2]{%
7763   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7764   \ifglxtrinsertinside\else##2\fi
7765 }%
7766 }

```

long-sc Backward compatibility:

```

7767 \@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.

```

7768 \newabbreviationstyle{long-noshort-sc-desc}%
7769 {%
7770   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
7771 }%
7772 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7773 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7774 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7775 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
7776 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7777 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

7778 \renewcommand*{\glxtrsubsequentfmt}[2]{%
7779   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7780   \ifglxtrinsertinside \else##2\fi
7781 }%
7782 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
7783   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7784   \ifglxtrinsertinside \else##2\fi
7785 }%
7786 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
7787   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7788   \ifglxtrinsertinside \else##2\fi
7789 }%
7790 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%

```

```

7791 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7792 \ifglxtrinsertinside \else##2\fi
7793 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

7794 \renewcommand*{\glxtrinlinefullformat}[2]{%
7795 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7796 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7797 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7798 }%
7799 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7800 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7801 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7802 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7803 }%
7804 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7805 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7806 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7807 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7808 }%
7809 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7810 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7811 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7812 \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7813 }%

```

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.

```

7814 \renewcommand*{\glxtrfullformat}[2]{%
7815 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7816 \ifglxtrinsertinside\else##2\fi
7817 }%
7818 \renewcommand*{\glxtrfullplformat}[2]{%
7819 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7820 \ifglxtrinsertinside\else##2\fi
7821 }%
7822 \renewcommand*{\Glsxtrfullformat}[2]{%
7823 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7824 \ifglxtrinsertinside\else##2\fi
7825 }%
7826 \renewcommand*{\Glsxtrfullplformat}[2]{%
7827 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7828 \ifglxtrinsertinside\else##2\fi
7829 }%
7830 }

```

long-desc-sc Backward compatibility:

```

7831 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}

```

ort-sc-footnote

```

7832 \newabbreviationstyle{short-sc-footnote}%
7833 {%
7834   \renewcommand*{\CustomAbbreviationFields}{%
7835     name={\glxtrfootnotename},
7836     sort={\the\glsshorttok},
7837     description={\the\glslongtok},%
7838     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
7839       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
7840       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7841     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7842       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
7843       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
7844     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

7845   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7846     \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7847     \glsattribute{\the\glslabeltok}{regular}%
7848     {%
7849       \glssetattribute{\the\glslabeltok}{regular}{false}%
7850     }%
7851   }%
7852 }%
7853 }%
7854 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7855   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7856   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7857   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
7858   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
7859   \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7860   \renewcommand*{\glxtrfullformat}[2]{%
7861     \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7862     \ifglxtrininsertinside\else##2\fi
7863     \protect\glxtrabbrvfootnote{##1}%
7864     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7865   }%
7866   \renewcommand*{\glxtrfullplformat}[2]{%
7867     \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7868     \ifglxtrininsertinside\else##2\fi
7869     \protect\glxtrabbrvfootnote{##1}%
7870     {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7871   }%
7872   \renewcommand*{\Glsxtrfullformat}[2]{%
7873     \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7874     \ifglxtrininsertinside\else##2\fi
7875     \protect\glxtrabbrvfootnote{##1}%

```

```

7876     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7877 }%
7878 \renewcommand*{\Glsxtrfullplformat}[2]{%
7879   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7880   \ifglxtrinsertinside\else##2\fi
7881   \protect\glxtrabbrvfootnote{##1}%
7882   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7883 }%

```

The first use full form and the inline full form use the short (long) style.

```

7884 \renewcommand*{\glsxtrinlinefullformat}[2]{%
7885   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7886   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7887   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7888 }%
7889 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
7890   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7891   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7892   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7893 }%
7894 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7895   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7896   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7897   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7898 }%
7899 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7900   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7901   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7902   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7903 }%
7904 }

```

footnote-sc Backward compatibility:

```

7905 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

7906 \newabbreviationstyle{short-sc-postfootnote}%
7907 {%
7908   \renewcommand*{\CustomAbbreviationFields}{%
7909     name={\glxtrfootnotename},
7910     sort={\the\glsshorttok},
7911     description={\the\glslongtok},%
7912     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
7913     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
7914     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.

```

7915 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7916   \csdef{glxtrpostlink\glscategorylabel}{%

```

```

7917 \glxtrifwasfirstuse
7918 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7919 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7920 {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
7921 }%
7922 {%
7923 }%
7924 \glshasattribute{\the\glslabeltok}{regular}%
7925 {%
7926 \glissetattribute{\the\glslabeltok}{regular}{false}%
7927 }%
7928 {}%
7929 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

7930 \renewcommand*{\glxtrsetupfulldefs}{%
7931 \let\glxtrifwasfirstuse\@secondoftwo
7932 }%
7933 }%
7934 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7935 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7936 \renewcommand*{\glabbrvfont}[1]{\glabbrvscfont{##1}}%
7937 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvscfont{##1}}%
7938 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongfootnotefont{##1}}%
7939 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

7940 \renewcommand*{\glxtrfullformat}[2]{%
7941 \glfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7942 \ifglxtrininsertinside\else##2\fi
7943 }%
7944 \renewcommand*{\glxtrfullplformat}[2]{%
7945 \glfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7946 \ifglxtrininsertinside\else##2\fi
7947 }%
7948 \renewcommand*{\Glsxtrfullformat}[2]{%
7949 \glfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7950 \ifglxtrininsertinside\else##2\fi
7951 }%
7952 \renewcommand*{\Glsxtrfullplformat}[2]{%
7953 \glfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7954 \ifglxtrininsertinside\else##2\fi
7955 }%

```

The first use full form and the inline full form use the short (long) style.

```

7956 \renewcommand*{\glxtrinlinefullformat}[2]{%
7957   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7958   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7959   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7960 }%
7961 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7962   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7963   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7964   \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7965 }%
7966 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7967   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7968   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7969   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7970 }%
7971 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7972   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7973   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7974   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7975 }%
7976 }

```

postfootnote-sc Backward compatibility:

```
7977 \@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.

```
7978 \newcommand*{\glxtrsmfont}[1]{\textsmaller{##1}}
```

`\glsabbrvsmfont` Added for consistent naming.

```
7979 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}
```

`\glxtrfirstsmfont` Maintained for backward compatibility.

```
7980 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{##1}}
```

`\glfirstabbrvsmfont` Added for consistent naming.

```
7981 \newcommand*{\glfirstabbrvsmfont}{\glxtrfirstsmfont}
```

and for the default short form suffix:

`\glxtrsmsuffix`

```
7982 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}
```

long-short-sm

```
7983 \newabbreviationstyle{long-short-sm}%
7984 {%
7985   \renewcommand*{\CustomAbbreviationFields}{%
7986     name={\glxtrlongshortname},
7987     sort={\the\glsshorttok},
7988     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7989       \protect\glxtrfullsep{\the\glslabeltok}%
7990       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
7991     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7992       \protect\glxtrfullsep{\the\glslabeltok}%
7993       \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
7994     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}},%
7995     description={\the\glslongtok}}%
7996   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7997     \glshasattribute{\the\glslabeltok}{regular}%
7998     {%
7999       \glissetattribute{\the\glslabeltok}{regular}{false}%
8000     }%
8001   }%
8002 }%
8003 }%
8004 {%
8005   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8006   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8007   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%

```

Use the default long fonts.

```
8008 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8009 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```
8010 \renewcommand*\glxtrfullformat[2]{%
8011   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8012   \ifglxtrininsertinside\else##2\fi
8013   \glxtrfullsep{##1}%
8014   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8015 }%
8016 \renewcommand*\glxtrfullplformat[2]{%
8017   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
8018   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8019   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8020 }%
8021 \renewcommand*\Glsxtrfullformat[2]{%
8022   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
8023   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8024   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8025 }%
8026 \renewcommand*\Glsxtrfullplformat[2]{%
8027   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%

```



```

8028 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8029 \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8030 }%
8031 }

```

g-short-sm-desc

```

8032 \newabbreviationstyle{long-short-sm-desc}%
8033 {%
8034 \renewcommand*{\CustomAbbreviationFields}{%
8035   name={\glxtrlongshortdescname},
8036   sort={\glxtrlongshortdescsort},%
8037   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8038     \protect\glxtrfullsep{\the\glslabeltok}%
8039     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8040   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8041     \protect\glxtrfullsep{\the\glslabeltok}%
8042     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8043   text={\protect\glssabbrvsmfont{\the\glsshorttok}},%
8044   plural={\protect\glssabbrvsmfont{\the\glsshortpltok}}}%
8045 }%

```

Unset the regular attribute if it has been set.

```

8046 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8047   \glshasattribute{\the\glslabeltok}{regular}%
8048   {%
8049     \glissetattribute{\the\glslabeltok}{regular}{false}%
8050   }%
8051   {}%
8052 }%
8053 }%
8054 {%

```

As long-short-sm style:

```

8055 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8056 }

```

short-sm-long Now the short (long) version

```

8057 \newabbreviationstyle{short-sm-long}%
8058 {%
8059 \renewcommand*{\CustomAbbreviationFields}{%
8060   name={\glxtrshortlongname},
8061   sort={\the\glsshorttok},
8062   description={\the\glslongtok},%
8063   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8064     \protect\glxtrfullsep{\the\glslabeltok}%
8065     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8066   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8067     \protect\glxtrfullsep{\the\glslabeltok}%
8068     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8069   plural={\protect\glssabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8070 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8071   \glshasattribute{\the\glslabeltok}{regular}%
8072   {%
8073     \glsselattribute{\the\glslabeltok}{regular}{false}%
8074   }%
8075   {}%
8076 }%
8077 }%
8078 {%
8079 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8080 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8081 \renewcommand*{\abbrvpluralsuffix}{\protect\glstrmsuffix}%
8082 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8083 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8084 \renewcommand*{\glsxtrfullformat}[2]{%
8085   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8086   \ifglsxtrininsertinside\else##2\fi
8087   \glsxtrfullsep{##1}%
8088   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8089 }%
8090 \renewcommand*{\glsxtrfullplformat}[2]{%
8091   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8092   \ifglsxtrininsertinside\else##2\fi
8093   \glsxtrfullsep{##1}%
8094   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8095 }%
8096 \renewcommand*{\Glsxtrfullformat}[2]{%
8097   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8098   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8099   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8100 }%
8101 \renewcommand*{\Glsxtrfullplformat}[2]{%
8102   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8103   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8104   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8105 }%
8106 }

```

rt-sm-long-desc As before but user provides description

```

8107 \newabbreviationstyle{short-sm-long-desc}%
8108 {%
8109   \renewcommand*{\CustomAbbreviationFields}{%
8110     name={\glsxtrshortlongdescname},
8111     sort={\glsxtrshortlongdescsort},
8112     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}}%
8113     \protect\glsxtrfullsep{\the\glslabeltok}%

```

```

8114     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8115     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8116     \protect\glxtrfullsep{\the\glslabeltok}%
8117     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8118     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8119     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8120 }%

```

Unset the regular attribute if it has been set.

```

8121 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8122     \glshasattribute{\the\glslabeltok}{regular}%
8123     {%
8124         \glssetattribute{\the\glslabeltok}{regular}{false}%
8125     }%
8126     {}%
8127 }%
8128 }%
8129 {%

```

As short-sm-long style:

```

8130 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8131 }

```

short-sm

```

8132 \newabbreviationstyle{short-sm}%
8133 {%
8134     \renewcommand*{\CustomAbbreviationFields}{%
8135         name={\glxtrshortnolongname},
8136         sort={\the\glsshorttok},
8137         first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8138         firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8139         text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8140         plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8141         description={\the\glslongtok}}%
8142     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8143         \glssetattribute{\the\glslabeltok}{regular}{true}}%
8144     }%
8145     {%
8146         \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8147         \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8148         \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8149         \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8150         \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8151 \renewcommand*{\glxtrinlinefullformat}[2]{%
8152     \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}%
8153     \ifglxtrininsertinside##2\fi}%
8154     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8155     \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%

```

```

8156 }%
8157 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8158   \protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}%
8159     \ifglxtrininsertinside##2\fi}%
8160   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8161   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8162 }%

8163 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8164   \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}%
8165     \ifglxtrininsertinside##2\fi}%
8166   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8167   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8168 }%

8169 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8170   \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}%
8171     \ifglxtrininsertinside##2\fi}%
8172   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8173   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8174 }%

```

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.

```

8175 \renewcommand*{\glxtrfullformat}[2]{%
8176   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8177   \ifglxtrininsertinside\else##2\fi
8178 }%
8179 \renewcommand*{\glxtrfullplformat}[2]{%
8180   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8181   \ifglxtrininsertinside\else##2\fi
8182 }%
8183 \renewcommand*{\Glsxtrfullformat}[2]{%
8184   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8185   \ifglxtrininsertinside\else##2\fi
8186 }%
8187 \renewcommand*{\Glsxtrfullplformat}[2]{%
8188   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8189   \ifglxtrininsertinside\else##2\fi
8190 }%
8191 }

```

short-sm-nolong

```
8192 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```

8193 \newabbreviationstyle{short-sm-desc}%
8194 {%
8195   \renewcommand*{\CustomAbbreviationFields}{%
8196     name={\glxtrshortdescname},

```

```

8197     sort={\the\glsshorttok},
8198     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8199     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8200     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8201     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8202     description={\the\glslongtok}}%
8203 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8204   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8205 }%
8206 {%
8207   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8208   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8209   \renewcommand*\abbrvpluralsuffix{\protect\glsxtrmsuffix}%
8210   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8211   \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8212   \renewcommand*\glsxtrinlinefullformat}[2]{%
8213     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8214     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8215     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8216   }%
8217   \renewcommand*\glsxtrinlinefullplformat}[2]{%
8218     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8219     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8220     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8221   }%
8222   \renewcommand*\Glsxtrinlinefullformat}[2]{%
8223     \glsfirstabbrvsmfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8224     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8225     \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8226   }%
8227   \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8228     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8229     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8230     \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8231   }%

```

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.

```

8232   \renewcommand*\glsxtrfullformat}[2]{%
8233     \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8234     \ifglsxtrininsertinside\else##2\fi
8235   }%
8236   \renewcommand*\glsxtrfullplformat}[2]{%
8237     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8238     \ifglsxtrininsertinside\else##2\fi
8239   }%
8240   \renewcommand*\Glsxtrfullformat}[2]{%
8241     \glsfirstabbrvsmfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%

```

```

8242     \ifglxtrinsertinside\else##2\fi
8243 }%
8244 \renewcommand*{\Glsxtrfullplformat}[2]{%
8245     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8246     \ifglxtrinsertinside\else##2\fi
8247 }%
8248 }

```

-sm-nolong-desc

```

8249 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

8250 \newabbreviationstyle{nolong-short-sm}%
8251 {%
8252     \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
8253 }%
8254 {%
8255     \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8256 \renewcommand*{\glxtrinlinefullformat}[2]{%
8257     \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8258         \ifglxtrinsertinside##2\fi}%
8259     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8260     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8261 }%
8262 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8263     \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8264         \ifglxtrinsertinside##2\fi}%
8265     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8266     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8267 }%
8268 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8269     \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8270         \ifglxtrinsertinside##2\fi}%
8271     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8272     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8273 }%
8274 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8275     \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8276         \ifglxtrinsertinside##2\fi}%
8277     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8278     \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8279 }%
8280 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8281 \newabbreviationstyle{long-noshort-sm}%

```

```

8282 {%
8283   \renewcommand*{\CustomAbbreviationFields}{%
8284     name={\glstrlongnosshortname},
8285     sort={\the\glssshorttok},
8286     first={\protect\glstrfirstlongdefaultfont{\the\glstrlongtok}},
8287     firstplural={\protect\glstrfirstlongdefaultfont{\the\glstrlongpltok}},
8288     text={\protect\glstrlongdefaultfont{\the\glstrlongtok}},
8289     plural={\protect\glstrlongdefaultfont{\the\glstrlongpltok}},%
8290     description={\the\glstrlongtok}%
8291   }%
8292   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8293     \glsssetAttribute{\the\glstrlabeltok}{regular}{true}}%
8294 }%
8295 {%
8296   \renewcommand*{\glstrabbrvfont}[1]{\glstrabbrvsmfont{##1}}%
8297   \renewcommand*{\glstrfirstabbrvfont}[1]{\glstrfirstabbrvsmfont{##1}}%
8298   \renewcommand*{\glstrabbrvpluralsuffix}{\protect\glstrsrmsuffix}%
8299   \renewcommand*{\glstrfirstlongfont}[1]{\glstrfirstlongdefaultfont{##1}}%
8300   \renewcommand*{\glstrlongfont}[1]{\glstrlongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8301   \renewcommand*{\glstrsubsequentfmt}[2]{%
8302     \glstrlongdefaultfont{\glstraccesslong{##1}\ifglstrinsertinside ##2\fi}%
8303     \ifglstrinsertinside \else##2\fi
8304   }%
8305   \renewcommand*{\glstrsubsequentplfmt}[2]{%
8306     \glstrlongdefaultfont{\glstraccesslongpl{##1}\ifglstrinsertinside ##2\fi}%
8307     \ifglstrinsertinside \else##2\fi
8308   }%
8309   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8310     \glstrlongdefaultfont{\Glsstraccesslong{##1}\ifglstrinsertinside ##2\fi}%
8311     \ifglstrinsertinside \else##2\fi
8312   }%
8313   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8314     \glstrlongdefaultfont{\Glsstraccesslongpl{##1}\ifglstrinsertinside ##2\fi}%
8315     \ifglstrinsertinside \else##2\fi
8316   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8317   \renewcommand*{\glstrinlinefullformat}[2]{%
8318     \glstrfirstlongdefaultfont{\glstraccesslong{##1}\ifglstrinsertinside##2\fi}%
8319     \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
8320     \glstrparen{\protect\glstrfirstabbrvsmfont{\glstraccessshort{##1}}}%
8321   }%
8322   \renewcommand*{\glstrinlinefullplformat}[2]{%
8323     \glstrfirstlongdefaultfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
8324     \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
8325     \glstrparen{\protect\glstrfirstabbrvsmfont{\glstraccessshortpl{##1}}}%
8326   }%
8327   \renewcommand*{\Glsxtrinlinefullformat}[2]{%

```

```

8328 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8329 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8330 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8331 }%
8332 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8333 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8334 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8335 \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8336 }%

```

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.

```

8337 \renewcommand*{\glxtrfullformat}[2]{%
8338 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8339 \ifglxtrinsertinside\else##2\fi
8340 }%
8341 \renewcommand*{\glxtrfullplformat}[2]{%
8342 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8343 \ifglxtrinsertinside\else##2\fi
8344 }%
8345 \renewcommand*{\Glsxtrfullformat}[2]{%
8346 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8347 \ifglxtrinsertinside\else##2\fi
8348 }%
8349 \renewcommand*{\Glsxtrfullplformat}[2]{%
8350 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8351 \ifglxtrinsertinside\else##2\fi
8352 }%
8353 }

```

long-sm Backward compatibility:

```

8354 \@glxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}

```

noshort-sm-desc The smaller font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8355 \newabbreviationstyle{long-noshort-sm-desc}%
8356 {%
8357 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8358 }%
8359 {%
8360 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8361 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8362 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
8363 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8364 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8365 \renewcommand*{\glxtrsubsequentfmt}[2]{%
8366 \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8367 \ifglxtrinsertinside \else##2\fi

```



```

8368 }%
8369 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
8370   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8371   \ifglxtrinsertinside \else##2\fi
8372 }%
8373 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8374   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8375   \ifglxtrinsertinside \else##2\fi
8376 }%
8377 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8378   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8379   \ifglxtrinsertinside \else##2\fi
8380 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8381 \renewcommand*{\glxtrinlinefullformat}[2]{%
8382   \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8383   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8384   \glxtrparen{\protect\glssfirstabbrvsmfont{\glssaccessshort{##1}}}%
8385 }%
8386 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8387   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8388   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8389   \glxtrparen{\protect\glssfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8390 }%
8391 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8392   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8393   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8394   \glxtrparen{\protect\glssfirstabbrvsmfont{\glssaccessshort{##1}}}%
8395 }%
8396 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8397   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8398   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8399   \glxtrparen{\protect\glssfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8400 }%

```

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.

```

8401 \renewcommand*{\glxtrfullformat}[2]{%
8402   \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8403   \ifglxtrinsertinside\else##2\fi
8404 }%
8405 \renewcommand*{\glxtrfullplformat}[2]{%
8406   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8407   \ifglxtrinsertinside\else##2\fi
8408 }%
8409 \renewcommand*{\Glsxtrfullformat}[2]{%
8410   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8411   \ifglxtrinsertinside\else##2\fi
8412 }%

```

```

8413 \renewcommand*{\Glsxtrfullplformat}[2]{%
8414   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8415   \ifglxtrinsertinside\else##2\fi
8416 }%
8417 }

```

long-desc-sm Backward compatibility:

```

8418 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

8419 \newabbreviationstyle{short-sm-footnote}%
8420 {%
8421   \renewcommand*{\CustomAbbreviationFields}{%
8422     name={\glsxtrfootnotename},
8423     sort={\the\glsshorttok},
8424     description={\the\glslongtok},%
8425     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8426       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8427       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8428     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8429       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8430       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8431     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.

```

8432 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8433   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8434   \glsattribute{\the\glslabeltok}{regular}%
8435   {%
8436     \glssetattribute{\the\glslabeltok}{regular}{false}%
8437   }%
8438   {}%
8439 }%
8440 }%
8441 {%
8442   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8443   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8444   \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8445   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8446   \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8447 \renewcommand*{\glsxtrfullformat}[2]{%
8448   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8449   \ifglxtrinsertinside\else##2\fi
8450   \protect\glxtrabbrvfootnote{##1}%
8451   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8452 }%
8453 \renewcommand*{\glsxtrfullplformat}[2]{%

```

```

8454 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8455 \ifglxtrinsertinside\else##2\fi
8456 \protect\glxtrabbrvfootnote{##1}%
8457 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8458 }%
8459 \renewcommand*{\Glsxtrfullformat}[2]{%
8460 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8461 \ifglxtrinsertinside\else##2\fi
8462 \protect\glxtrabbrvfootnote{##1}%
8463 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8464 }%
8465 \renewcommand*{\Glsxtrfullplformat}[2]{%
8466 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8467 \ifglxtrinsertinside\else##2\fi
8468 \protect\glxtrabbrvfootnote{##1}%
8469 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8470 }%

```

The first use full form and the inline full form use the short (long) style.

```

8471 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8472 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8473 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8474 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8475 }%
8476 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8477 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8478 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8479 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8480 }%
8481 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8482 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8483 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8484 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8485 }%
8486 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8487 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8488 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8489 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8490 }%
8491 }

```

footnote-sm Backward compatibility:

```

8492 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

8493 \newabbreviationstyle{short-sm-postfootnote}%
8494 {%
8495 \renewcommand*{\CustomAbbreviationFields}{%
8496 name={\glxtrfootnotename},
8497 sort={\the\glsshorttok},

```

```

8498   description={\the\glslongtok},%
8499   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
8500   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
8501   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.

```

8502   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8503     \csdef{glsxtrpostlink\glscategorylabel}{%
8504       \glsxtrifwasfirstuse
8505       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8506         \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}}%
8507         {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
8508     }%
8509   }%
8510 }%
8511 \glshasattribute{\the\glslabeltok}{regular}%
8512 {%
8513   \glssetattribute{\the\glslabeltok}{regular}{false}%
8514 }%
8515 {}%
8516 }%

```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```

8517   \renewcommand*{\glsxtrsetupfulldefs}{%
8518     \let\glsxtrifwasfirstuse\@secondoftwo
8519   }%
8520 }%
8521 {%
8522   \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8523   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8524   \renewcommand*\{abbrvpluralsuffix}{\protect\glsxtrmsuffix}%
8525   \renewcommand*\{glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8526   \renewcommand*\{glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8527   \renewcommand*\{glsxtrfullformat}[2]{%
8528     \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8529     \ifglsxtrininsertinside\else##2\fi
8530   }%
8531   \renewcommand*\{glsxtrfullplformat}[2]{%
8532     \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8533     \ifglsxtrininsertinside\else##2\fi
8534   }%
8535   \renewcommand*\{Glsxtrfullformat}[2]{%
8536     \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8537     \ifglsxtrininsertinside\else##2\fi

```

```

8538 }%
8539 \renewcommand*{\Glsxtrfullplformat}[2]{%
8540   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8541   \ifglxtrinsertinside\else##2\fi
8542 }%

```

The first use full form and the inline full form use the short (long) style.

```

8543 \renewcommand*{\glxtrinlinefullformat}[2]{%
8544   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8545   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8546   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8547 }%
8548 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8549   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8550   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8551   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8552 }%
8553 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8554   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8555   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8556   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8557 }%
8558 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8559   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8560   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8561   \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8562 }%
8563 }

```

postfootnote-sm Backward compatibility:

```

8564 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

8565 \newcommand*{\glsabbrvemfont}[1]{\emph{##1}}%

```

`\glsfirstabbrvemfont`

```

8566 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{##1}}%

```

The default short form suffix:

`\glxtremsuffix`

```

8567 \newcommand*{\glxtremsuffix}{\glxtrabbrvpluralsuffix}

```

`\glsfirstlongemfont` Only used by the “long-em” styles.

```

8568 \newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{##1}}%

```

`\glslongemfont` Only used by the “long-em” styles.

```
8569 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`long-short-em` The long form is just set in the default long font.

```
8570 \newabbreviationstyle{long-short-em}%
8571 {%
8572   \renewcommand*{\CustomAbbreviationFields}{%
8573     name={\glsxtrlongshortname},
8574     sort={\the\glsshorttok},
8575     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8576       \protect\glsxtrfullsep{\the\glslabeltok}%
8577       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
8578     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8579       \protect\glsxtrfullsep{\the\glslabeltok}%
8580       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
8581     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
8582     description={\the\glslongtok}}%
8583   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8584     \glsattribute{\the\glslabeltok}{regular}%
8585     {%
8586       \glssetattribute{\the\glslabeltok}{regular}{false}%
8587     }%
8588   }%
8589 }%
8590 }%
8591 {%
8592   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
8593   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
8594   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%

```

Use the default long fonts.

```
8595 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8596 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8597 \renewcommand*{\glsxtrfullformat}[2]{%
8598   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8599   \ifglsxtrininsertinside\else##2\fi
8600   \glsxtrfullsep{##1}%
8601   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
8602 }%
8603 \renewcommand*{\glsxtrfullplformat}[2]{%
8604   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8605   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8606   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
8607 }%
8608 \renewcommand*{\Glsxtrfullformat}[2]{%
8609   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8610   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8611   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%

```

```

8612 }%
8613 \renewcommand*{\Glsxtrfullplformat}[2]{%
8614   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8615   \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8616   \glsxtrparen{\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
8617 }%
8618 }

```

g-short-em-desc

```

8619 \newabbreviationstyle{long-short-em-desc}%
8620 {%
8621   \renewcommand*{\CustomAbbreviationFields}{%
8622     name={\glsxtrlongshortdescname},
8623     sort={\glsxtrlongshortdescsort},%
8624     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8625       \protect\glsxtrfullsep{\the\glslabeltok}%
8626       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
8627     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8628       \protect\glsxtrfullsep{\the\glslabeltok}%
8629       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
8630     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
8631     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
8632   }%

```

Unset the regular attribute if it has been set.

```

8633 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8634   \glshasattribute{\the\glslabeltok}{regular}%
8635   {%
8636     \glissetattribute{\the\glslabeltok}{regular}{false}%
8637   }%
8638   {}}%
8639 }%
8640 }%
8641 {%

```

As long-short-em style:

```

8642 \GlsXtrUseAbbrStyleFmts{long-short-em}%
8643 }

```

ong-em-short-em

```

8644 \newabbreviationstyle{long-em-short-em}%
8645 {%
8646   \glslongemfont is used in the description since \glsdesc doesn't set the style.
8647   \renewcommand*{\CustomAbbreviationFields}{%
8648     name={\glsxtrlongshortname},
8649     sort={\the\glsshorttok},
8650     first={\protect\glsfirstlongemfont{\the\glslongtok}%
8651       \protect\glsxtrfullsep{\the\glslabeltok}%
8652       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%

```

```

8652   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}}%
8653   \protect\glsxtrfullsep{\the\glslabeltok}}%
8654   \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

8655   plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
8656   description={\protect\glsfirstlongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

8657   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8658     \glshasattribute{\the\glslabeltok}{regular}}%
8659   {%
8660     \glssetattribute{\the\glslabeltok}{regular}{false}}%
8661   }%
8662   {}%
8663 }%
8664 }%
8665 {%
8666   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
8667   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
8668   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
8669   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
8670   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8671   \renewcommand*{\glsxtrfullformat}[2]{%
8672     \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8673     \ifglsxtrinsertinside\else##2\fi
8674     \glsxtrfullsep{##1}}%
8675     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
8676   }%
8677   \renewcommand*{\glsxtrfullplformat}[2]{%
8678     \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8679     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}}%
8680     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
8681   }%
8682   \renewcommand*{\Glsxtrfullformat}[2]{%
8683     \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8684     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}}%
8685     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
8686   }%
8687   \renewcommand*{\Glsxtrfullplformat}[2]{%
8688     \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8689     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}}%
8690     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
8691   }%
8692 }

```

m-short-em-desc

```

8693 \newabbreviationstyle{long-em-short-em-desc}%
8694 {%

```



```

8695 \renewcommand*{\CustomAbbreviationFields}{%
8696   name={\glxtrlongshortdescname},
8697   sort={\glxtrlongshortdescsort},%
8698   first={\protect\glsfirstlongemfont{\the\glslongtok}%
8699     \protect\glxtrfullsep{\the\glslabeltok}%
8700     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
8701   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
8702     \protect\glxtrfullsep{\the\glslabeltok}%
8703     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
8704   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
8705   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
8706 }%

```

Unset the regular attribute if it has been set.

```

8707 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8708   \glshasattribute{\the\glslabeltok}{regular}%
8709   {%
8710     \glissetattribute{\the\glslabeltok}{regular}{false}%
8711   }%
8712   {}%
8713 }%
8714 }%
8715 {%
8716   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
8717 }

```

short-em-long Now the short (long) version

```

8718 \newabbreviationstyle{short-em-long}%
8719 {%
8720   \renewcommand*{\CustomAbbreviationFields}{%
8721     name={\glxtrshortlongname},
8722     sort={\the\glsshorttok},
8723     description={\the\glslongtok},%
8724     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
8725       \protect\glxtrfullsep{\the\glslabeltok}%
8726       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8727     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
8728       \protect\glxtrfullsep{\the\glslabeltok}%
8729       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8730     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%

```

Unset the regular attribute if it has been set.

```

8731 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8732   \glshasattribute{\the\glslabeltok}{regular}%
8733   {%
8734     \glissetattribute{\the\glslabeltok}{regular}{false}%
8735   }%
8736   {}%
8737 }%
8738 }%

```

8739 {%

Mostly as short-long style:

```
8740 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
8741 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
8742 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
8743 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8744 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8745 \renewcommand*{\glsxtrfullformat}[2]{%
8746   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
8747   \ifglsxtrinsertinside\else##2\fi
8748   \glsxtrfullsep{##1}%
8749   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8750 }%
8751 \renewcommand*{\glsxtrfullplformat}[2]{%
8752   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
8753   \ifglsxtrinsertinside\else##2\fi
8754   \glsxtrfullsep{##1}%
8755   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8756 }%
8757 \renewcommand*{\Glsxtrfullformat}[2]{%
8758   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
8759   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8760   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8761 }%
8762 \renewcommand*{\Glsxtrfullplformat}[2]{%
8763   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
8764   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8765   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8766 }%
8767 }
```

rt-em-long-desc As before but user provides description

```
8768 \newabbreviationstyle{short-em-long-desc}%
8769 {%
8770   \renewcommand*{\CustomAbbreviationFields}{%
8771     name={\glsxtrshortlongdescname},
8772     sort={\glsxtrshortlongdescsort},
8773     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
8774     \protect\glsxtrfullsep{\the\glslabeltok}%
8775     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8776     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}}%
8777     \protect\glsxtrfullsep{\the\glslabeltok}%
8778     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8779     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
8780     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
8781 }
```

Unset the regular attribute if it has been set.

```

8782 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8783   \glshasattribute{\the\glslabeltok}{regular}%
8784   {%
8785     \glissetattribute{\the\glslabeltok}{regular}{false}%
8786   }%
8787   {}%
8788 }%
8789 }%
8790 {%
8791   \GlsXtrUseAbbrStyleFmts{short-em-long}%
8792 }

```

hort-em-long-em

```

8793 \newabbreviationstyle{short-em-long-em}%
8794 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

8795 \renewcommand*{\CustomAbbreviationFields}{%
8796   name={\glsxtrshortlongname},
8797   sort={\the\glsshorttok},
8798   description={\protect\glslongemfont{\the\glslongtok}},%
8799   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
8800     \protect\glsxtrfullsep{\the\glslabeltok}%
8801     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
8802   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
8803     \protect\glsxtrfullsep{\the\glslabeltok}%
8804     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
8805   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}

```

Unset the regular attribute if it has been set.

```

8806 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8807   \glshasattribute{\the\glslabeltok}{regular}%
8808   {%
8809     \glissetattribute{\the\glslabeltok}{regular}{false}%
8810   }%
8811   {}%
8812 }%
8813 }%
8814 {%
8815 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
8816 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
8817 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
8818 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
8819 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8820 \renewcommand*{\glsxtrfullformat}[2]{%
8821   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsetinside##2\fi}%
8822   \ifglsxtrinsetinside\else##2\fi
8823   \glsxtrfullsep{##1}%

```

```

8824 \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
8825 }%
8826 \renewcommand*{\glsxtrfullplformat}[2]{%
8827 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8828 \ifglsxtrininsertinside\else##2\fi
8829 \glsxtrfullsep{##1}%
8830 \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
8831 }%
8832 \renewcommand*{\Glsxtrfullformat}[2]{%
8833 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8834 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8835 \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
8836 }%
8837 \renewcommand*{\Glsxtrfullplformat}[2]{%
8838 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8839 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8840 \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
8841 }%
8842 }

```

em-long-em-desc

```

8843 \newabbreviationstyle{short-em-long-em-desc}%
8844 {%
8845 \renewcommand*{\CustomAbbreviationFields}{%
8846 name={\glsxtrshortlongdescname},%
8847 sort={\glsxtrshortlongdescsort},%
8848 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
8849 \protect\glsxtrfullsep{\the\glslabeltok}%
8850 \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
8851 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
8852 \protect\glsxtrfullsep{\the\glslabeltok}%
8853 \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
8854 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
8855 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
8856 }%

```

Unset the regular attribute if it has been set.

```

8857 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8858 \glshasattribute{\the\glslabeltok}{regular}%
8859 {%
8860 \glissetattribute{\the\glslabeltok}{regular}{false}%
8861 }%
8862 {}%
8863 }%
8864 }%
8865 {%
8866 \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
8867 }

```

short-em

```

8868 \newabbreviationstyle{short-em}%
8869 {%
8870   \renewcommand*{\CustomAbbreviationFields}{%
8871     name={\glxtrshortnolongname},
8872     sort={\the\glsshorttok},
8873     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
8874     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
8875     text={\protect\glsabbrvemfont{\the\glsshorttok}},
8876     plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
8877     description={\the\glslongtok}}%
8878   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8879     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8880 }%
8881 {%
8882   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
8883   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
8884   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
8885   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8886   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8887   \renewcommand*{\glxtrinlinefullformat}[2]{%
8888     \protect\glsfirstabbrvemfont{\glsaccessshort{##1}}%
8889     \ifglxtrininsertinside##2\fi}%
8890   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8891   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8892 }%
8893   \renewcommand*{\glxtrinlinefullplformat}[2]{%
8894     \protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}%
8895     \ifglxtrininsertinside##2\fi}%
8896   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8897   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8898 }%
8899   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8900     \protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}%
8901     \ifglxtrininsertinside##2\fi}%
8902   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8903   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8904 }%
8905   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8906     \protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}%
8907     \ifglxtrininsertinside##2\fi}%
8908   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8909   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8910 }%

```

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.

```

8911   \renewcommand*{\glxtrfullformat}[2]{%

```

```

8912 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8913 \ifglxtrinsertinside\else##2\fi
8914 }%
8915 \renewcommand*{\glxtrfullplformat}[2]{%
8916 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8917 \ifglxtrinsertinside\else##2\fi
8918 }%
8919 \renewcommand*{\Glsxtrfullformat}[2]{%
8920 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8921 \ifglxtrinsertinside\else##2\fi
8922 }%
8923 \renewcommand*{\Glsxtrfullplformat}[2]{%
8924 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8925 \ifglxtrinsertinside\else##2\fi
8926 }%
8927 }

```

short-em-nolong

```

8928 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

8929 \newabbreviationstyle{short-em-desc}%
8930 {%
8931 \renewcommand*{\CustomAbbreviationFields}{%
8932 name={\glxtrshortdescname},
8933 sort={\the\glsshorttok},
8934 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
8935 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
8936 text={\protect\glsabbrvemfont{\the\glsshorttok}},
8937 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
8938 description={\the\glslongtok}}%
8939 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8940 \glssetattribute{\the\glslabeltok}{regular}{true}}%
8941 }%
8942 {%
8943 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
8944 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
8945 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
8946 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8947 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8948 \renewcommand*{\glxtrinlinefullformat}[2]{%
8949 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8950 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8951 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8952 }%
8953 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8954 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8955 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

8956 \glstrparen{\glfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8957 }%
8958 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8959 \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8960 \ifglxtrinsertinside\else##2\fi\glstrfullsep{##1}%
8961 \glstrparen{\glfirstlongdefaultfont{\glsaccesslong{##1}}}%
8962 }%
8963 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8964 \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8965 \ifglxtrinsertinside\else##2\fi\glstrfullsep{##1}%
8966 \glstrparen{\glfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8967 }%

```

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.

```

8968 \renewcommand*{\glstrfullformat}[2]{%
8969 \glfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8970 \ifglxtrinsertinside\else##2\fi
8971 }%
8972 \renewcommand*{\glstrfullplformat}[2]{%
8973 \glfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8974 \ifglxtrinsertinside\else##2\fi
8975 }%
8976 \renewcommand*{\Glsxtrfullformat}[2]{%
8977 \glfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8978 \ifglxtrinsertinside\else##2\fi
8979 }%
8980 \renewcommand*{\Glsxtrfullplformat}[2]{%
8981 \glfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8982 \ifglxtrinsertinside\else##2\fi
8983 }%
8984 }

```

-em-nolong-desc

```

8985 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

8986 \newabbreviationstyle{nolong-short-em}%
8987 {%
8988 \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
8989 }%
8990 {%
8991 \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8992 \renewcommand*{\glxtrinlinefullformat}[2]{%
8993 \protect\glfirstlongdefaultfont{\glsaccesslong{##1}%
8994 \ifglxtrinsertinside##2\fi}%
8995 \ifglxtrinsertinside\else##2\fi\glstrfullsep{##1}%
8996 \glstrparen{\glfirstabbrvemfont{\glsaccessshort{##1}}}%

```

```

8997 }%
8998 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8999   \protect\glslongdefaultfont{\glssaccesslongpl{##1}%
9000   \ifglxtrininsertinside##2\fi}%
9001   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9002   \glxtrparen{\glslongdefaultfont{\glssaccessshortpl{##1}}}%
9003 }%
9004 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9005   \protect\glslongdefaultfont{\Glsaccesslong{##1}%
9006   \ifglxtrininsertinside##2\fi}%
9007   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9008   \glxtrparen{\glslongdefaultfont{\Glsaccessshort{##1}}}%
9009 }%
9010 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9011   \protect\glslongdefaultfont{\Glsaccesslongpl{##1}%
9012   \ifglxtrininsertinside##2\fi}%
9013   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9014   \glxtrparen{\glslongdefaultfont{\Glsaccessshortpl{##1}}}%
9015 }%
9016 }

```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9017 \newabbreviationstyle{long-noshort-em}%
9018 {%
9019   \renewcommand*{\CustomAbbreviationFields}{%
9020     name={\glxtrlongnoshortname},
9021     sort={\the\glsshorttok},
9022     first={\protect\glslongdefaultfont{\the\glslongtok}},
9023     firstplural={\protect\glslongdefaultfont{\the\glslongpltok}},
9024     text={\protect\glslongdefaultfont{\the\glslongtok}},
9025     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9026     description={\the\glslongtok}%
9027   }%
9028   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9029     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9030 }%
9031 {%
9032   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9033   \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9034   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
9035   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
9036   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9037 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9038   \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9039   \ifglxtrininsertinside \else##2\fi
9040 }%
9041 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9042   \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%

```



```

9043 \ifglxtrinsertinside \else##2\fi
9044 }%
9045 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9046 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9047 \ifglxtrinsertinside \else##2\fi
9048 }%
9049 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9050 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9051 \ifglxtrinsertinside \else##2\fi
9052 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9053 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9054 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9055 \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
9056 \Glsxtrparen{\protect\Glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9057 }%
9058 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9059 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9060 \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
9061 \Glsxtrparen{\protect\Glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9062 }%
9063 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9064 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9065 \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
9066 \Glsxtrparen{\protect\Glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9067 }%
9068 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9069 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9070 \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
9071 \Glsxtrparen{\protect\Glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9072 }%

```

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.

```

9073 \renewcommand*{\Glsxtrfullformat}[2]{%
9074 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9075 \ifglxtrinsertinside\else##2\fi
9076 }%
9077 \renewcommand*{\Glsxtrfullplformat}[2]{%
9078 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9079 \ifglxtrinsertinside\else##2\fi
9080 }%
9081 \renewcommand*{\Glsxtrfullformat}[2]{%
9082 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9083 \ifglxtrinsertinside\else##2\fi
9084 }%
9085 \renewcommand*{\Glsxtrfullplformat}[2]{%
9086 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9087 \ifglxtrinsertinside\else##2\fi

```

```

9088 }%
9089 }

```

long-em Backward compatibility:

```

9090 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9091 \newabbreviationstyle{long-em-noshort-em}%
9092 {%
9093   \renewcommand*{\CustomAbbreviationFields}{%
9094     name={\glxtrlongnoshortname},
9095     sort={\the\glsshorttok},
9096     first={\protect\glslongemfont{\the\glslongtok}},
9097     firstplural={\protect\glslongemfont{\the\glslongpltok}},
9098     text={\protect\glslongemfont{\the\glslongtok}},
9099     plural={\protect\glslongemfont{\the\glslongpltok}},%
9100     description={\protect\glslongemfont{\the\glslongtok}}%
9101   }%
9102   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9103     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9104 }%
9105 {%
9106   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9107   \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
9108   \renewcommand*{\glslfirstabbrvfont}[1]{\glslfirstabbrvemfont{##1}}%
9109   \renewcommand*{\glslfirstlongfont}[1]{\glslfirstlongemfont{##1}}%
9110   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9111 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9112   \glslongemfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9113   \ifglxtrininsertinside \else##2\fi
9114 }%
9115 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9116   \glslongemfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9117   \ifglxtrininsertinside \else##2\fi
9118 }%
9119 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9120   \glslongemfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9121   \ifglxtrininsertinside \else##2\fi
9122 }%
9123 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9124   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9125   \ifglxtrininsertinside \else##2\fi
9126 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9127 \renewcommand*{\glxtrinlinefullformat}[2]{%
9128   \glslfirstlongemfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9129   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9130 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9131 }%
9132 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9133 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9134 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9135 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9136 }%
9137 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9138 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9139 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9140 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9141 }%
9142 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9143 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9144 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
9145 \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9146 }%

```

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.

```

9147 \renewcommand*{\glsxtrfullformat}[2]{%
9148 \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9149 \ifglsxtrinsertinside\else##2\fi
9150 }%
9151 \renewcommand*{\glsxtrfullplformat}[2]{%
9152 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9153 \ifglsxtrinsertinside\else##2\fi
9154 }%
9155 \renewcommand*{\Glsxtrfullformat}[2]{%
9156 \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
9157 \ifglsxtrinsertinside\else##2\fi
9158 }%
9159 \renewcommand*{\Glsxtrfullplformat}[2]{%
9160 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
9161 \ifglsxtrinsertinside\else##2\fi
9162 }%
9163 }

```

`noshort-em-noreg` Like `long-em-noshort-em` but doesn't set the regular attribute.

```

9164 \newabbreviationstyle{long-em-noshort-em-noreg}%
9165 {%
9166 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%

```

Unset the regular attribute if it has been set.

```

9167 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9168 \glshasattribute{\the\glslabeltok}{regular}%
9169 {%
9170 \glssetattribute{\the\glslabeltok}{regular}{false}%
9171 }%
9172 }%

```

```

9173 }%
9174 }%
9175 {%
9176 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
9177 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glssshort`.

```

9178 \newabbreviationstyle{long-noshort-em-desc}%
9179 {%
9180 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9181 }%
9182 {%
9183 \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
9184 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9185 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
9186 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
9187 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9188 \renewcommand*{\glssxtrsubsequentfmt}[2]{%
9189 \glsslongdefaultfont{\glssaccesslong{##1}\ifglssxtrininsertinside ##2\fi}%
9190 \ifglssxtrininsertinside \else##2\fi
9191 }%
9192 \renewcommand*{\glssxtrsubsequentplfmt}[2]{%
9193 \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglssxtrininsertinside ##2\fi}%
9194 \ifglssxtrininsertinside \else##2\fi
9195 }%
9196 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9197 \glsslongdefaultfont{\Glsaccesslong{##1}\ifglssxtrininsertinside ##2\fi}%
9198 \ifglssxtrininsertinside \else##2\fi
9199 }%
9200 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9201 \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglssxtrininsertinside ##2\fi}%
9202 \ifglssxtrininsertinside \else##2\fi
9203 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9204 \renewcommand*{\glssxtrinlinefullformat}[2]{%
9205 \glssfirstlongdefaultfont{\glssaccesslong{##1}\ifglssxtrininsertinside##2\fi}%
9206 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9207 \glssxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshort{##1}}}%
9208 }%
9209 \renewcommand*{\glssxtrinlinefullplformat}[2]{%
9210 \glssfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglssxtrininsertinside##2\fi}%
9211 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
9212 \glssxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9213 }%
9214 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9215 \glssfirstlongdefaultfont{\Glsaccesslong{##1}\ifglssxtrininsertinside##2\fi}%

```

```

9216     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9217     \glxtrparen{\protect\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
9218 }%
9219 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9220     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9221     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9222     \glxtrparen{\protect\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
9223 }%

```

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.

```

9224 \renewcommand*{\glxtrfullformat}[2]{%
9225     \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9226     \ifglxtrinsertinside\else##2\fi
9227 }%
9228 \renewcommand*{\glxtrfullplformat}[2]{%
9229     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9230     \ifglxtrinsertinside\else##2\fi
9231 }%
9232 \renewcommand*{\Glsxtrfullformat}[2]{%
9233     \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9234     \ifglxtrinsertinside\else##2\fi
9235 }%
9236 \renewcommand*{\Glsxtrfullplformat}[2]{%
9237     \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9238     \ifglxtrinsertinside\else##2\fi
9239 }%
9240 }

```

long-desc-em Backward compatibility:

```

9241 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}

```

noshort-em-desc The short form is explicitly invoked through commands like `\glssshort`. The long form is emphasized.

```

9242 \newabbreviationstyle{long-em-noshort-em-desc}%
9243 {%
9244     \renewcommand*{\CustomAbbreviationFields}{%
9245         name={\glxtrlongnoshortdescname},
9246         sort={\the\glslongtok},
9247         first={\protect\glsfirstlongemfont{\the\glslongtok}},
9248         firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9249         text={\glslongemfont{\the\glslongtok}},
9250         plural={\glslongemfont{\the\glslongpltok}}%
9251     }%
9252     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9253         \glsssetattribute{\the\glslabeltok}{regular}{true}}%
9254 }%
9255 {%
9256     \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%

```

```

9257 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9258 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9259 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
9260 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9261 \renewcommand*\glsxtrsubsequentfmt}[2]{%
9262   \glslongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9263   \ifglsxtrininsertinside \else##2\fi
9264 }%
9265 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
9266   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9267   \ifglsxtrininsertinside \else##2\fi
9268 }%
9269 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
9270   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9271   \ifglsxtrininsertinside \else##2\fi
9272 }%
9273 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
9274   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9275   \ifglsxtrininsertinside \else##2\fi
9276 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9277 \renewcommand*\glsxtrinlinefullformat}[2]{%
9278   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9279   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9280   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9281 }%
9282 \renewcommand*\glsxtrinlinefullplformat}[2]{%
9283   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9284   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9285   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9286 }%
9287 \renewcommand*\Glsxtrinlinefullformat}[2]{%
9288   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9289   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9290   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9291 }%
9292 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
9293   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9294   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9295   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9296 }%

```

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.

```

9297 \renewcommand*\glsxtrfullformat}[2]{%
9298   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9299   \ifglsxtrininsertinside\else##2\fi
9300 }%

```

```

9301 \renewcommand*{\glxtrfullplformat}[2]{%
9302   \glsfirstlongemfont{\glaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9303   \ifglxtrininsertinside\else##2\fi
9304 }%
9305 \renewcommand*{\Glsxtrfullformat}[2]{%
9306   \glsfirstlongemfont{\glaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9307   \ifglxtrininsertinside\else##2\fi
9308 }%
9309 \renewcommand*{\Glsxtrfullplformat}[2]{%
9310   \glsfirstlongemfont{\glaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9311   \ifglxtrininsertinside\else##2\fi
9312 }%
9313 }

```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```

9314 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
9315 {%
9316   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%
   Unset the regular attribute if it has been set.
9317   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9318     \glshasattribute{\the\glslabeltok}{regular}%
9319     {%
9320       \glissetattribute{\the\glslabeltok}{regular}{false}%
9321     }%
9322   }%
9323 }%
9324 }%
9325 {%
9326   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
9327 }

```

ort-em-footnote

```

9328 \newabbreviationstyle{short-em-footnote}%
9329 {%
9330   \renewcommand*{\CustomAbbreviationFields}{%
9331     name={\glxtrfootnotename},
9332     sort={\the\glsshorttok},
9333     description={\the\gslongtok},%
9334     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9335       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9336       {\protect\glsfirstlongfootnotefont{\the\gslongtok}}},%
9337     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9338       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9339       {\protect\glsfirstlongfootnotefont{\the\gslongpltok}}},%
9340     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.
9341   \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

9342 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9343 \glsattribute{\the\glslabeltok}{regular}%
9344 {%
9345 \glssetattribute{\the\glslabeltok}{regular}{false}%
9346 }%
9347 {}%
9348 }%
9349 }%
9350 {%
9351 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9352 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9353 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9354 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9355 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9356 \renewcommand*{\glsxtrfullformat}[2]{%
9357 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9358 \ifglsxtrininsertinside\else##2\fi
9359 \protect\glsxtrabbrvfootnote{##1}%
9360 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9361 }%
9362 \renewcommand*{\glsxtrfullplformat}[2]{%
9363 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9364 \ifglsxtrininsertinside\else##2\fi
9365 \protect\glsxtrabbrvfootnote{##1}%
9366 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9367 }%
9368 \renewcommand*{\Glsxtrfullformat}[2]{%
9369 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9370 \ifglsxtrininsertinside\else##2\fi
9371 \protect\glsxtrabbrvfootnote{##1}%
9372 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9373 }%
9374 \renewcommand*{\Glsxtrfullplformat}[2]{%
9375 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9376 \ifglsxtrininsertinside\else##2\fi
9377 \protect\glsxtrabbrvfootnote{##1}%
9378 {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9379 }%

```

The first use full form and the inline full form use the short (long) style.

```

9380 \renewcommand*{\glsxtrininlinefullformat}[2]{%
9381 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9382 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9383 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9384 }%
9385 \renewcommand*{\glsxtrininlinefullplformat}[2]{%
9386 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9387 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%

```



```

9388 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9389 }%
9390 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9391 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9392 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9393 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9394 }%
9395 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9396 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9397 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9398 \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
9399 }%
9400 }

```

footnote-em Backward compatibility:

```

9401 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

9402 \newabbreviationstyle{short-em-postfootnote}%
9403 {%
9404 \renewcommand*{\CustomAbbreviationFields}{%
9405 name={\glxtrfootnotename},
9406 sort={\the\glsshorttok},
9407 description={\the\glslongtok},%
9408 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
9409 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
9410 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

9411 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9412 \csdef{glxtrpostlink\glscategorylabel}{%
9413 \glxtrifwasfirstuse
9414 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9415 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9416 {\glsfirstlongfootnotefont{\glssentrylong{\glslabel}}}}%
9417 }%
9418 {}%
9419 }%
9420 \glshasattribute{\the\glslabeltok}{regular}%
9421 {%
9422 \glissetattribute{\the\glslabeltok}{regular}{false}%
9423 }%
9424 {}%
9425 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9426 \renewcommand*\glxtrsetupfulldefs}{%
9427 \let\glxtrifwasfirstuse\@secondoftwo
9428 }%
9429 }%
9430 {%
9431 \renewcommand*\abbrvpluralsuffix{\protect\glxxtremsuffix}%
9432 \renewcommand\glxabbrvfont[1]{\glxabbrvemfont{##1}}%
9433 \renewcommand*\glxfirstabbrvfont[1]{\glxfirstabbrvemfont{##1}}%
9434 \renewcommand*\glxfirstlongfont[1]{\glxfirstlongfootnotefont{##1}}%
9435 \renewcommand*\glxlongfont[1]{\glxlongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9436 \renewcommand*\glxtrfullformat}[2]{%
9437 \glxfirstabbrvemfont{\glxaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9438 \ifglxtrininsertinside\else##2\fi
9439 }%
9440 \renewcommand*\glxtrfullplformat}[2]{%
9441 \glxfirstabbrvemfont{\glxaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9442 \ifglxtrininsertinside\else##2\fi
9443 }%
9444 \renewcommand*\Glxtrfullformat}[2]{%
9445 \glxfirstabbrvemfont{\Glxaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9446 \ifglxtrininsertinside\else##2\fi
9447 }%
9448 \renewcommand*\Glxtrfullplformat}[2]{%
9449 \glxfirstabbrvemfont{\Glxaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9450 \ifglxtrininsertinside\else##2\fi
9451 }%

```

The first use full form and the inline full form use the short (long) style.

```

9452 \renewcommand*\glxtrininlinefullformat}[2]{%
9453 \glxfirstabbrvemfont{\glxaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9454 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9455 \glxtrparen{\glxfirstlongfootnotefont{\glxaccesslong{##1}}}%
9456 }%
9457 \renewcommand*\glxtrininlinefullplformat}[2]{%
9458 \glxfirstabbrvemfont{\glxaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9459 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9460 \glxtrparen{\glxfirstlongfootnotefont{\glxaccesslongpl{##1}}}%
9461 }%
9462 \renewcommand*\Glxtrininlinefullformat}[2]{%
9463 \glxfirstabbrvemfont{\Glxaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9464 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9465 \glxtrparen{\glxfirstlongfootnotefont{\glxaccesslong{##1}}}%
9466 }%
9467 \renewcommand*\Glxtrininlinefullplformat}[2]{%
9468 \glxfirstabbrvemfont{\Glxaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9469 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9470 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9471 }%
9472 }

```

postfootnote-em Backward compatibility:

```

9473 \@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.

```

9474 \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.

```

9475 \ifdef\glscurrentfieldvalue
9476 {
9477 \newcommand*{\glsxtruserparen}[2]{%
9478 \glsxtrfullsep{#2}%
9479 \glsxtrparen
9480 {#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%
9481 }
9482 }
9483 {
9484 \newcommand*{\glsxtruserparen}[2]{%
9485 \glsxtrfullsep{#2}%
9486 \glsxtrparen
9487 {#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{}}%
9488 }
9489 }

```

Font used for short form:

lsabbrvuserfont

```

9490 \newcommand*{\glsabbrvuserfont}[1]{\glsabbrvdefaultfont{#1}}

```

Font used for short form on first use:

stabbrvuserfont

```

9491 \newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}

```

Font used for long form:

glslonguserfont

```

9492 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}

```

Font used for long form on first use:

rstlonguserfont

```
9493 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
9494 \newcommand*{\glsxtrusersuffix}{\glsxtrabbrvpluralsuffix}
```

long-short-user

```
9495 \newabbreviationstyle{long-short-user}%
9496 {%
9497   \renewcommand*{\CustomAbbreviationFields}{%
9498     name={\glsxtrlongshortname},
9499     sort={\the\glsshorttok},
9500     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
9501       \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%
9502     {\the\glslabeltok}},%
9503     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
9504       \protect\glsxtruserparen
9505         {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
9506     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9507     description={\protect\glslonguserfont{\the\glslongtok}}}%
9508 }
```

Unset the regular attribute if it has been set.

```
9508 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9509   \glshasattribute{\the\glslabeltok}{regular}%
9510   {%
9511     \glssetattribute{\the\glslabeltok}{regular}{false}%
9512   }%
9513   {}%
9514 }%
9515 }%
9516 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
9517 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
9518 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
9519 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
9520 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
9521 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9522 \renewcommand*{\glsxtrfullformat}[2]{%
9523   \glsfirstlonguserfont{\glsaccesslong{##1}}\ifglxtrinsertinside##2\fi}%
9524   \ifglxtrinsertinside\else##2\fi
9525   \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
9526 }%
9527 \renewcommand*{\glsxtrfullplformat}[2]{%
9528   \glsfirstlonguserfont{\glsaccesslongpl{##1}}\ifglxtrinsertinside##2\fi}%
9529   \ifglxtrinsertinside\else##2\fi
```

```

9530 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9531 }%
9532 \renewcommand*{\Glsxtrfullformat}[2]{%
9533 \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9534 \ifglxtrinsertinside\else##2\fi
9535 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
9536 }%
9537 \renewcommand*{\Glsxtrfullplformat}[2]{%
9538 \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9539 \ifglxtrinsertinside\else##2\fi
9540 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9541 }%
9542 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

9543 \newabbreviationstyle{long-postshort-user}%
9544 {%
9545 \renewcommand*{\CustomAbbreviationFields}{%
9546 name={\glxtrlongshortname},
9547 sort={\the\glsshorttok},
9548 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
9549 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
9550 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9551 description={\protect\glslonguserfont{\the\glslongtok}}}%
9552 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9553 \csdef{glxtrpostlink\glscategorylabel}{%
9554 \glxtrifwasfirstuse
9555 {%
9556 \glxtruserparen
9557 {\glsfirstabbrvuserfont{\glstryshort{\glslabel}}}%
9558 {\glslabel}%
9559 }%
9560 }%
9561 }%
9562 \glshasattribute{\the\glslabeltok}{regular}%
9563 {%
9564 \glssetattribute{\the\glslabeltok}{regular}{false}%
9565 }%
9566 }%
9567 }%
9568 }%
9569 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

9570 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
9571 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
9572 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
9573 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
9574 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```
9575 \renewcommand*{\glxtrfullformat}[2]{%
9576   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9577   \ifglxtrinsertinside\else##2\fi
9578 }%
9579 \renewcommand*{\glxtrfullplformat}[2]{%
9580   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9581   \ifglxtrinsertinside\else##2\fi
9582 }%
9583 \renewcommand*{\Glsxtrfullformat}[2]{%
9584   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9585   \ifglxtrinsertinside\else##2\fi
9586 }%
9587 \renewcommand*{\Glsxtrfullplformat}[2]{%
9588   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9589   \ifglxtrinsertinside\else##2\fi
9590 }%
```

In-line format:

```
9591 \renewcommand*{\glxtrinlinefullformat}[2]{%
9592   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9593   \ifglxtrinsertinside\else##2\fi
9594   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
9595 }%
9596 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9597   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9598   \ifglxtrinsertinside\else##2\fi
9599   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9600 }%
9601 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9602   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9603   \ifglxtrinsertinside\else##2\fi
9604   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
9605 }%
9606 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9607   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9608   \ifglxtrinsertinside\else##2\fi
9609   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9610 }%
9611 }
```

ortuserdesname

```
9612 \newcommand*{\glxtrlongshortuserdesname}{%
9613   \protect\glslonguserfont{\the\glslongtok}%
9614   \protect\glxtruserparen
9615   {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
9616 }
```

short-user-desc Like long-postshort-user but the user supplies the description.

```

9617 \newabbreviationstyle{long-postshort-user-desc}%
9618 {%
9619   \renewcommand*{\CustomAbbreviationFields}{%
9620     name={\glxtrlongshortuserdescname},
9621     sort={\the\glslongtok},
9622     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
9623     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

9624     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
9625     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%
9626   }%
9627   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9628     \csdef{glxtrpostlink\glscategorylabel}{%
9629       \glxtrifwasfirstuse
9630       {%
9631         \glxtruserparen
9632         {\glsfirstabbrvuserfont{\glstryshort{\glslabel}}}%
9633         {\glslabel}%
9634       }%
9635     }%
9636   }%
9637   \glshasattribute{\the\glslabeltok}{regular}%
9638   {%
9639     \glssetattribute{\the\glslabeltok}{regular}{false}%
9640   }%
9641   {}%
9642 }%
9643 }%
9644 {%
9645   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
9646 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

9647 \newabbreviationstyle{short-postlong-user}%
9648 {%
9649   \renewcommand*{\CustomAbbreviationFields}{%
9650     name={\glxtrshortlongname},
9651     sort={\the\glsshorttok},
9652     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
9653     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

9654     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9655     description={\protect\glslonguserfont{\the\glslongtok}}%
9656   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9657     \csdef{glxtrpostlink\glscategorylabel}{%
9658       \glxtrifwasfirstuse
9659       {%
9660         \glxtruserparen
9661         {\glsfirstlonguserfont{\glstrylong{\glslabel}}}%
9662         {\glslabel}%

```

```

9663     }%
9664     {}%
9665     }%
9666     \glshasattribute{\the\glslabeltok}{regular}%
9667     {%
9668         \glissetattribute{\the\glslabeltok}{regular}{false}%
9669     }%
9670     {}%
9671 }%
9672 }%
9673 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

9674 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
9675 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
9676 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
9677 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
9678 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

9679 \renewcommand*{\glsxtrfullformat}[2]{%
9680     \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9681     \ifglsxtrininsertinside\else##2\fi
9682 }%
9683 \renewcommand*{\glsxtrfullplformat}[2]{%
9684     \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9685     \ifglsxtrininsertinside\else##2\fi
9686 }%
9687 \renewcommand*{\Glsxtrfullformat}[2]{%
9688     \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9689     \ifglsxtrininsertinside\else##2\fi
9690 }%
9691 \renewcommand*{\Glsxtrfullplformat}[2]{%
9692     \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9693     \ifglsxtrininsertinside\else##2\fi
9694 }%

```

In-line format:

```

9695 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9696     \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9697     \ifglsxtrininsertinside\else##2\fi
9698     \glsxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
9699 }%
9700 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9701     \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9702     \ifglsxtrininsertinside\else##2\fi
9703     \glsxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
9704 }%
9705 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9706     \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%

```



```

9707 \ifglxtrinsertinside\else##2\fi
9708 \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
9709 }%
9710 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9711 \glsfirstabbruserfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9712 \ifglxtrinsertinside\else##2\fi
9713 \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
9714 }%
9715 }

```

onguserdescname

```

9716 \newcommand*{\glxtrshortlonguserdescname}{%
9717 \protect\glssabbruserfont{\the\glssshorttok}%
9718 \protect\glxtruserparen
9719 {\protect\glslonguserfont{\the\glslongpltok}}}%
9720 {\the\glslabeltok}%
9721 }

```

long-user-desc Like short-postlong-user but leaves the user to specify the description.

```

9722 \newabbreviationstyle{short-postlong-user-desc}%
9723 {%
9724 \renewcommand*{\CustomAbbreviationFields}{%
9725 name={\glxtrshortlonguserdescname},
9726 sort={\the\glssshorttok},
9727 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
9728 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
9729 text={\protect\glssabbruserfont{\the\glssshorttok}},%
9730 plural={\protect\glssabbruserfont{\the\glssshortpltok}}}%
9731 }%
9732 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9733 \csdef{glxtrpostlink\glscategorylabel}{%
9734 \glxtrifwasfirstuse
9735 {%
9736 \glxtruserparen
9737 {\glsfirstlonguserfont{\glssentrylong{\glslabel}}}%
9738 {\glslabel}%
9739 }%
9740 {}%
9741 }%
9742 \glshasattribute{\the\glslabeltok}{regular}%
9743 {%
9744 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
9745 }%
9746 {}%
9747 }%
9748 }%
9749 {%
9750 \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
9751 }

```

short-user-desc

```
9752 \newabbreviationstyle{long-short-user-desc}%
9753 {%
9754   \renewcommand*{\CustomAbbreviationFields}{%
9755     name={\glxtrlongshortuserdescname},
9756     sort={\glxtrlongshortdescsort},%

9757     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
9758       \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
9759       {\the\glslabeltok}},%
9760     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
9761       \protect\glxtruserparen
9762       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
9763     text={\protect\glsabbrvfont{\the\glsshorttok}},%
9764     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
9765   }%

   Unset the regular attribute if it has been set.
9766   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9767     \glshasattribute{\the\glslabeltok}{regular}%
9768     {%
9769       \glissetattribute{\the\glslabeltok}{regular}{false}%
9770     }%
9771   }%
9772 }%
9773 }%
9774 {%
9775   \GlsXtrUseAbbrStyleFmts{long-short-user}%
9776 }
```

short-long-user

```
9777 \newabbreviationstyle{short-long-user}%
9778 {%

   \glslonguserfont is used in the description since \glsdesc doesn't set the style.
9779   \renewcommand*{\CustomAbbreviationFields}{%
9780     name={\glxtrshortlongname},
9781     sort={\the\glsshorttok},
9782     description={\protect\glslonguserfont{\the\glslongtok}},%
9783     first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
9784       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
9785       {\the\glslabeltok}},%
9786     firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
9787       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
9788       {\the\glslabeltok}},%

9789     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

   Unset the regular attribute if it has been set.
9790   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```

9791 \glshasattribute{\the\glslabelltok}{regular}%
9792 {%
9793 \glissetattribute{\the\glslabelltok}{regular}{false}%
9794 }%
9795 {}%
9796 }%
9797 }%
9798 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

9799 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
9800 \renewcommand*{\glssabrvfont[1]{\glssabrvuserfont{##1}}}%
9801 \renewcommand*{\glssfirstabrvfont}[1]{\glssfirstabrvuserfont{##1}}%
9802 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlonguserfont{##1}}%
9803 \renewcommand*{\glsslongfont}[1]{\glsslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9804 \renewcommand*{\glsxtrfullformat}[2]{%
9805 \glssfirstabrvuserfont{\glssaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
9806 \ifglsxtrinsertinside\else##2\fi
9807 \glsxtruserparen{\glssfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
9808 }%
9809 \renewcommand*{\glsxtrfullplformat}[2]{%
9810 \glssfirstabrvuserfont{\glssaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
9811 \ifglsxtrinsertinside\else##2\fi
9812 \glsxtruserparen{\glssfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
9813 }%
9814 \renewcommand*{\Glsxtrfullformat}[2]{%
9815 \glssfirstabrvuserfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
9816 \ifglsxtrinsertinside\else##2\fi
9817 \glsxtruserparen{\glssfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
9818 }%
9819 \renewcommand*{\Glsxtrfullplformat}[2]{%
9820 \glssfirstabrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
9821 \ifglsxtrinsertinside\else##2\fi
9822 \glsxtruserparen{\glssfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
9823 }%
9824 }

```

-long-user-desc

```

9825 \newabbreviationstyle{short-long-user-desc}%
9826 {%
9827 \renewcommand*{\CustomAbbreviationFields}{%
9828 name={\glsxtrshortlonguserdescname},
9829 sort={\glsxtrshortlongdescsort},%
9830 first={\protect\glssfirstabrvuserfont{\the\glssshorttok}}%
9831 \protect\glsxtruserparen{\protect\glssfirstlonguserfont{\the\glsslongtok}}%
9832 {\the\glslabelltok}},%
9833 firstplural={\protect\glssfirstabrvuserfont{\the\glssshortpltok}}%

```

```

9834 \protect\glstruserparen{\protect\glslonguserfont{\the\glslongpltok}}}%
9835 {\the\glslabeltok}},%
9836 text={\protect\glssabrvfont{\the\glsshorttok}},%
9837 plural={\protect\glssabrvfont{\the\glsshortpltok}}}%
9838 }%

```

Unset the regular attribute if it has been set.

```

9839 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9840 \glshasattribute{\the\glslabeltok}{regular}%
9841 {%
9842 \glissetattribute{\the\glslabeltok}{regular}{false}%
9843 }%
9844 {}%
9845 }%
9846 }%
9847 {%
9848 \GlsXtrUseAbbrStyleFmts{short-long-user}%
9849 }

```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the `markwords` attribute. They check if the inserted material (provided by the final optional argument of commands like `\gls`) starts with a hyphen. If it does, the insert is added to the parenthetical material. Note that commands like `\glstrlong` set `\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.

```

9850 \newrobustcmd*{\glstrifhyphenstart}[3]{%
9851 \ifx\glsinsert#1\relax
9852 \expandafter@\glstrifhyphenstart#1\relax\relax
9853 \@end@glstrifhyphenstart{#2}{#3}%
9854 \else
9855 \@glstrifhyphenstart#1\relax\relax\@end@glstrifhyphenstart{#2}{#3}%
9856 \fi
9857 }

```

`trifhyphenstart`

```

9858 \def\@glstrifhyphenstart#1#2\@end@glstrifhyphenstart#3#4{%
9859 \ifx-#1\relax#3\else #4\fi
9860 }

```

`rlonghyphenshort`

```
\glstrlonghyphenshort{<label>}{<long>}{<short>}{<insert>}
```

The `<long>` and `<short>` arguments may be the plural form. The `<long>` argument may also be the first letter uppercase form.

9861 \newcommand*{\glxtrlonghyphenshort}[4]{%

Grouping is needed to localise the redefinitions.

9862 {%

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.

9863 \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{}}%

9864 \glsfirslonghyphenfont{#2\ifglxtrininsertinside{#4}\fi}%

9865 \ifglxtrininsertinside\else{#4}\fi

9866 \glxtrfullsep{#1}%

9867 \glxtrparen{\glsfirstabbrvhyphenfont{#3\ifglxtrininsertinside{#4}\fi}%

9868 \ifglxtrininsertinside\else{#4}\fi}%

9869 }%

9870 }

abbrvhyphenfont

9871 \newcommand*{\glsabrvhyphenfont}{\glsabrvdefaultfont}%

abbrvhyphenfont

9872 \newcommand*{\glsfirstabbrvhyphenfont}{\glsabrvhyphenfont}%

slonghyphenfont

9873 \newcommand*{\glslonghyphenfont}{\glslongdefaultfont}%

tlonghyphenfont

9874 \newcommand*{\glsfirslonghyphenfont}{\glslonghyphenfont}%

The default short form suffix:

xtrhyphensuffix

9875 \newcommand*{\glxtrhyphensuffix}{\glxtrabbrvpluralsuffix}

en-short-hyphen Designed for use with the markwords attribute.

9876 \newabbreviationstyle{long-hyphen-short-hyphen}%

9877 {%

9878 \renewcommand*{\CustomAbbreviationFields}{%

9879 name={\glxtrlongshortname},

9880 sort={\the\glsshorttok},

9881 first={\protect\glsfirslonghyphenfont{\the\glslongtok}}%

9882 \protect\glxtrfullsep{\the\glslabeltok}}%

9883 \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%

9884 firstplural={\protect\glsfirslonghyphenfont{\the\glslongpltok}}%

9885 \protect\glxtrfullsep{\the\glslabeltok}}%

9886 \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%

9887 plural={\protect\glsabrvhyphenfont{\the\glsshortpltok}}},%

9888 description={\protect\glslonghyphenfont{\the\glslongtok}}}%

Unset the regular attribute if it has been set.

```

9889 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9890   \glshasattribute{\the\glslabeltok}{regular}%
9891   {%
9892     \glissetattribute{\the\glslabeltok}{regular}{false}%
9893   }%
9894   {}%
9895 }%
9896 }%
9897 {%
9898 \renewcommand*{\abbrvpluralsuffix}{\glsxtrhyphensuffix}%
9899 \renewcommand*{\glssabrvfont}[1]{\glssabrvhyphenfont{##1}}%
9900 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvhyphenfont{##1}}%
9901 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlonghyphenfont{##1}}%
9902 \renewcommand*{\glsslongfont}[1]{\glsslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9903 \renewcommand*{\glsxtrfullformat}[2]{%
9904   \glsxtrlonghyphenshort{##1}{\glssaccesslong{##1}}{\glssaccessshort{##1}}{##2}%
9905 }%
9906 \renewcommand*{\glsxtrfullplformat}[2]{%
9907   \glsxtrlonghyphenshort{##1}{\glssaccesslongpl{##1}}%
9908   {\glssaccessshortpl{##1}}{##2}%
9909 }%
9910 \renewcommand*{\Glsxtrfullformat}[2]{%
9911   \glsxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\glssaccessshort{##1}}{##2}%
9912 }%
9913 \renewcommand*{\Glsxtrfullplformat}[2]{%
9914   \glsxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
9915   {\glssaccessshortpl{##1}}{##2}%
9916 }%
9917 }

```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

9918 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
9919 {%
9920   \renewcommand*{\CustomAbbreviationFields}{%
9921     name={\glsxtrlongshortdescname},
9922     sort={\glsxtrlongshortdescsort},
9923     first={\protect\glssfirstlonghyphenfont{\the\glsslongtok}}%
9924     \protect\glsxtrfullsep{\the\glslabeltok}%
9925     \glsxtrparen{\protect\glssfirstabbrvhyphenfont{\the\glssshorttok}}},%
9926     firstplural={\protect\glssfirstlonghyphenfont{\the\glsslongpltok}}%
9927     \protect\glsxtrfullsep{\the\glslabeltok}%
9928     \glsxtrparen{\protect\glssfirstabbrvhyphenfont{\the\glssshortpltok}}},%
9929     text={\protect\glssabrvhyphenfont{\the\glssshorttok}},%
9930     plural={\protect\glssabrvhyphenfont{\the\glssshortpltok}}}%
9931   }%

```

Unset the regular attribute if it has been set.

```

9932 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9933   \glshasattribute{\the\glslabeltok}{regular}%
9934   {%
9935     \glissetattribute{\the\glslabeltok}{regular}{false}%
9936   }%
9937   {}%
9938 }%
9939 }%
9940 {%
9941   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
9942 }

```

onghyphennoshort

```
\glsxtrlonghyphennoshort{<label>}{<long>}{<insert>}
```

```

9943 \newcommand*{\glsxtrlonghyphennoshort}[3]{%

```

Grouping is needed to localise the redefinitions.

```
9944 {%
```

If *<insert>* starts with a hyphen, redefine `\glsxtrwordsep` 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 `\glsxtrwordsep` if *<insert>* doesn't start with a hyphen.

```

9945   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{%
9946     \glsfirstlonghyphenfont{#2\ifglsxtrininsertinside{#3}\fi}%
9947     \ifglsxtrininsertinside\else{#3}\fi
9948   }%
9949 }

```

short-desc-noreg

This version doesn't show the short form (except explicitly with `\glsxtrshort`). Since `\glsxtrshort` 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.

```

9950 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
9951 {%
9952   \renewcommand*{\CustomAbbreviationFields}{%
9953     name={\glsxtrlongnoshortdescname},
9954     sort={\expandonce\glxtrorglong},
9955     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
9956     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
9957     plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
9958   }%

```

Unset the regular attribute if it has been set.

```

9959 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9960   \glshasattribute{\the\glslabeltok}{regular}%
9961   {%
9962     \glissetattribute{\the\glslabeltok}{regular}{false}%

```

```

9963 }%
9964 {}%
9965 }%
9966 }%
9967 {%
9968 \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

9969 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
9970 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
9971 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
9972 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonghyphenfont{##1}}%
9973 \renewcommand*\glslongfont[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9974 \renewcommand*\glxtrsubsequentfmt[2]{%
9975   \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
9976 }%
9977 \renewcommand*\glxtrsubsequentplfmt[2]{%
9978   \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
9979 }%
9980 \renewcommand*\Glsxtrsubsequentfmt[2]{%
9981   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
9982 }%
9983 \renewcommand*\Glsxtrsubsequentplfmt[2]{%
9984   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
9985 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9986 \renewcommand*\glxtrininlinefullformat[2]{%
9987   \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
9988   \glxtrfullsep{##1}%
9989   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
9990 }%
9991 \renewcommand*\glxtrininlinefullplformat[2]{%
9992   \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
9993   \glxtrfullsep{##1}%
9994   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
9995 }%
9996 \renewcommand*\Glsxtrininlinefullformat[2]{%
9997   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
9998   \glxtrfullsep{##1}%
9999   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
10000 }%
10001 \renewcommand*\Glsxtrininlinefullplformat[2]{%
10002   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10003   \glxtrfullsep{##1}%
10004   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10005 }%

```

The first use full form only displays the long form.


```

10006 \renewcommand*{\glxtrfullformat}[2]{%
10007   \glxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
10008 }%
10009 \renewcommand*{\glxtrfullplformat}[2]{%
10010   \glxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
10011 }%
10012 \renewcommand*{\Glsxtrfullformat}[2]{%
10013   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10014 }%
10015 \renewcommand*{\Glsxtrfullplformat}[2]{%
10016   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10017 }%
10018 }

```

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.

```

10019 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10020 {%
10021   \renewcommand*{\CustomAbbreviationFields}{%
10022     name={\glxtrlongnoshortname},
10023     sort={\the\glsshorttok},
10024     first={\protect\glxtrlonghyphenfont{\the\glslongtok}},%
10025     firstplural={\protect\glxtrlonghyphenfont{\the\glslongpltok}},%
10026     text={\protect\glxtrlonghyphenfont{\the\glslongtok}},%
10027     plural={\protect\glxtrlonghyphenfont{\the\glslongpltok}},%
10028     description={\the\glslongtok}%
10029   }%

```

Unset the regular attribute if it has been set.

```

10030 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10031   \glshasattribute{\the\glslabeltok}{regular}%
10032   {%
10033     \glissetattribute{\the\glslabeltok}{regular}{false}%
10034   }%
10035   {}}%
10036 }%
10037 }%
10038 {%
10039   \GlsXtrUseAbbrStyleFmts{long-desc}%
10040 }

```

glxtrlonghyphen `\glxtrlonghyphen{<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.

```

10041 \newcommand*{\glxtrlonghyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10042 {%
10043   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}}%
10044   \glsfirslonghyphenfont{#1}%
10045 }%
10046 }

```

posthyphenshort `\glxtrposthyphenshort{<label>}{<insert>}`

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\glxtrlonghyphenshort` but omits the *<long>* part. This always uses the singular short form.

```

10047 \newcommand*{\glxtrposthyphenshort}[2]{%
10048   {%
10049     \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}}%
10050     \ifglxtrininsertinside{\glsfirslonghyphenfont{#2}}\else{#2}\fi
10051     \glxtrfullsep{#1}%
10052     \glxtrparen
10053     {\glsfirstabbrvhyphenfont{\glstryshort{#1}}\ifglxtrininsertinside{#2}\fi}%
10054     \ifglxtrininsertinside\else{#2}\fi
10055   }%
10056 }%
10057 }

```

hyphensubsequent `\glxtrposthyphensubsequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```

10058 \newcommand*{\glxtrposthyphensubsequent}[2]{%
10059   \glssabrvfont{\ifglxtrininsertinside {#2}\fi}%
10060   \ifglxtrininsertinside \else{#2}\fi
10061 }

```

postshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10062 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10063 {%
10064   \renewcommand*{\CustomAbbreviationFields}{%
10065     name={\glxtrlongshortname},
10066     sort={\the\glsshorttok},
10067     first={\protect\glsfirslonghyphenfont{\the\glslongtok}},%
10068     firstplural={\protect\glsfirslonghyphenfont{\the\glslongpltok}},%
10069     plural={\protect\glssabrvhyphenfont{\the\glsshortpltok}},%
10070     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10071   \renewcommand*{\GlsXtrPostNewAbbreviation}{%

```

```

10072 \csdef{glxstrpostlink\glscategorylabel}{%
10073 \glxstrifwasfirstuse
10074 {%
10075 \glxstrposthyphenshort{\glslabel}{\glsininsert}%
10076 }%
10077 {%

```

Put the insertion into the post-link:

```

10078 \glxstrposthyphensubsequent{\glslabel}{\glsininsert}%
10079 }%
10080 }%
10081 \glshasattribute{\the\glslabeltok}{regular}%
10082 {%
10083 \glissetattribute{\the\glslabeltok}{regular}{false}%
10084 }%
10085 {}%
10086 }%
10087 }%
10088 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10089 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10090 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10091 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10092 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10093 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10094 \renewcommand*{\glxtrssubsequentfmt}[2]{%
10095 \glsabbrvfont{\glssaccessshort{##1}}%
10096 }%
10097 \renewcommand*{\glxtrssubsequentplfmt}[2]{%
10098 \glsabbrvfont{\glssaccessshortpl{##1}}%
10099 }%
10100 \renewcommand*{\Glsxtrssubsequentfmt}[2]{%
10101 \glsabbrvfont{\Glsaccessshort{##1}}%
10102 }%
10103 \renewcommand*{\Glsxtrssubsequentplfmt}[2]{%
10104 \glsabbrvfont{\Glsaccessshortpl{##1}}%
10105 }%

```

First use full form:

```

10106 \renewcommand*{\glxtrfullformat}[2]{%
10107 \glxtrlonghyphen{\glssaccesslong{##1}}{##1}{##2}%
10108 }%
10109 \renewcommand*{\glxtrfullplformat}[2]{%
10110 \glxtrlonghyphen{\glssaccesslongpl{##1}}{##1}{##2}%
10111 }%
10112 \renewcommand*{\Glsxtrfullformat}[2]{%
10113 \glxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10114 }%

```

```

10115 \renewcommand*{\Glsxtrfullplformat}[2]{%
10116   \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10117 }%

```

In-line format.

```

10118 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10119   \glsfirstlonghyphenfont{\Glsaccesslong{##1}%
10120     \ifglsxtrininsertinside{##2}\fi}%
10121   \ifglsxtrininsertinside \else{##2}\fi
10122 }%
10123 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10124   \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}%
10125     \ifglsxtrininsertinside{##2}\fi}%
10126   \ifglsxtrininsertinside \else{##2}\fi
10127 }%
10128 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10129   \glsfirstlonghyphenfont{\Glsaccesslong{##1}%
10130     \ifglsxtrininsertinside{##2}\fi}%
10131   \ifglsxtrininsertinside \else{##2}\fi
10132 }%
10133 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10134   \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}%
10135     \ifglsxtrininsertinside{##2}\fi}%
10136   \ifglsxtrininsertinside \else{##2}\fi
10137 }%
10138 }

```

ort-hyphen-desc Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

10139 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10140 {%
10141   \renewcommand*{\CustomAbbreviationFields}{%
10142     name={\glsxtrlongshortdescname},
10143     sort={\glsxtrlongshortdescsort},%
10144     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10145     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10146     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10147     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10148 }%
10149 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10150   \csdef{glsxtrpostlink\glscategorylabel}{%
10151     \glsxtrifwasfirstuse
10152     {%
10153       \glsxtrposthyphenshort{\glslabel}{\glsinsert}%
10154     }%
10155     {%

```

Put the insertion into the post-link:

```

10156       \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10157     }%
10158   }%

```

```

10159 \glshasattribute{\the\glslabeltok}{regular}%
10160 {%
10161 \glissetattribute{\the\glslabeltok}{regular}{false}%
10162 }%
10163 {}%
10164 }%
10165 }%
10166 {%
10167 \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10168 }

```

`\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.

```

10169 \newcommand*{\glxtrshorthyphenlong}[4]{%

```

Grouping is needed to localise the redefinitions.

```

10170 {%

```

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.)

```

10171 \glxtrifhyphenstart{#4}{\def\glxtrwordsep{-}}{%
10172 \glsfirstabbrvhyphenfont{#2\ifglxtrininsertinside{#4}\fi}%
10173 \ifglxtrininsertinside\else{#4}\fi
10174 \glxtrfullsep{#1}%
10175 \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrininsertinside{#4}\fi}%
10176 \ifglxtrininsertinside\else{#4}\fi}%
10177 }%
10178 }

```

`hen-long-hyphen` Designed for use with the `markwords` attribute.

```

10179 \newabbreviationstyle{short-hyphen-long-hyphen}%
10180 {%
10181 \renewcommand*{\CustomAbbreviationFields}{%
10182 name={\glxtrshortlongname},
10183 sort={\the\glsshorttok},
10184 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10185 \protect\glxtrfullsep{\the\glslabeltok}%
10186 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10187 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10188 \protect\glxtrfullsep{\the\glslabeltok}%
10189 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10190 plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}},%
10191 description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10192 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10193   \glshasattribute{\the\glslabeltok}{regular}%
10194   {%
10195     \glsselattribute{\the\glslabeltok}{regular}{false}%
10196   }%
10197   {}%
10198 }%
10199 }%
10200 {%
10201 \renewcommand*{\abbrvpluralsuffix}{\glsxtrhyphensuffix}%
10202 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10203 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10204 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10205 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10206 \renewcommand*{\glsxtrfullformat}[2]{%
10207   \glsxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10208 }%
10209 \renewcommand*{\glsxtrfullplformat}[2]{%
10210   \glsxtrshorthyphenlong{##1}%
10211   {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10212 }%
10213 \renewcommand*{\Glsxtrfullformat}[2]{%
10214   \glsxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10215 }%
10216 \renewcommand*{\Glsxtrfullplformat}[2]{%
10217   \glsxtrshorthyphenlong{##1}%
10218   {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
10219 }%
10220 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

10221 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
10222 {%
10223   \renewcommand*{\CustomAbbreviationFields}{%
10224     name={\glsxtrshortlongdescname},
10225     sort={\glsxtrshortlongdescsort},
10226     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}%
10227     \protect\glsxtrfullsep{\the\glslabeltok}%
10228     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10229     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}%
10230     \protect\glsxtrfullsep{\the\glslabeltok}%
10231     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10232     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10233     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10234 }%

```

Unset the regular attribute if it has been set.

```

10235 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10236   \glshasattribute{\the\glslabeltok}{regular}%
10237   {%
10238     \glissetattribute{\the\glslabeltok}{regular}{false}%
10239   }%
10240   {}%
10241 }%
10242 }%
10243 {%
10244   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
10245 }

```

`\glstrshorthyphen` `\glstrshorthyphen{<short>}{<label>}{<insert>}`

Used by short-hyphen-postlong-hyphen. The `<insert>` is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

10246 \newcommand*{\glstrshorthyphen}[3]{%
    Grouping is needed to localise the redefinitions.
10247   {%
10248     \glstrifhyphenstart{#3}{\def\glstrwordsep{-}}}%
10249     \glsfirstabbrvhyphenfont{#1}%
10250   }%
10251 }

```

`\glstrposthyphenlong` `\glstrposthyphenlong{<label>}{<insert>}`

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glstrshorthyphenlong` but omits the `<short>` part. This always uses the singular long form.

```

10252 \newcommand*{\glstrposthyphenlong}[2]{%
10253   {%
10254     \glstrifhyphenstart{#2}{\def\glstrwordsep{-}}}%
10255     \ifglstrinsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
10256     \glstrfullsep{#1}%
10257     \glstrparen
10258     {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglstrinsertinside{#2}\fi}%
10259     \ifglstrinsertinside\else{#2}\fi
10260   }%
10261 }%
10262 }

```

`postlong-hyphen` Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10263 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
10264 {%
10265   \renewcommand*{\CustomAbbreviationFields}{%
10266     name={\glxtrshortlongname},
10267     sort={\the\glsshorttok},
10268     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10269     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10270     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
10271     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10272 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10273   \csdef{glxtrpostlink\glscategorylabel}{%
10274     \glxtrifwasfirstuse
10275     {%
10276       \glxtrposthyphenlong{\glslabel}{\glsinsert}%
10277     }%
10278   }%

```

Put the insertion into the post-link:

```

10279     \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10280   }%
10281 }%
10282 \glshasattribute{\the\glslabeltok}{regular}%
10283 {%
10284   \glssetattribute{\the\glslabeltok}{regular}{false}%
10285 }%
10286 {}%
10287 }%
10288 }%
10289 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10290 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10291 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10292 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10293 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10294 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10295 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10296   \glsabbrvfont{\glsaccessshort{##1}}%
10297 }%
10298 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10299   \glsabbrvfont{\glsaccessshortpl{##1}}%
10300 }%
10301 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10302   \glsabbrvfont{\Glsaccessshort{##1}}%
10303 }%
10304 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10305   \glsabbrvfont{\Glsaccessshortpl{##1}}%
10306 }%

```


First use full form:

```

10307 \renewcommand*{\glxstrfullformat}[2]{%
10308   \glxstrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
10309 }%
10310 \renewcommand*{\glxstrfullplformat}[2]{%
10311   \glxstrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
10312 }%
10313 \renewcommand*{\Glsxstrfullformat}[2]{%
10314   \glxstrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
10315 }%
10316 \renewcommand*{\Glsxstrfullplformat}[2]{%
10317   \glxstrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
10318 }%

```

In-line format. Commands like `\glxstrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

10319 \renewcommand*{\glxstrinlinefullformat}[2]{%
10320   \glsfirstabbrvhyphenfont{\glsaccessshort{##1}}%
10321   \ifglxstrinsertinside{##2}\fi}%
10322   \ifglxstrinsertinside \else{##2}\fi
10323 }%
10324 \renewcommand*{\glxstrinlinefullplformat}[2]{%
10325   \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}}%
10326   \ifglxstrinsertinside{##2}\fi}%
10327   \ifglxstrinsertinside \else{##2}\fi
10328 }%
10329 \renewcommand*{\Glsxstrinlinefullformat}[2]{%
10330   \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}}%
10331   \ifglxstrinsertinside{##2}\fi}%
10332   \ifglxstrinsertinside \else{##2}\fi
10333 }%
10334 \renewcommand*{\Glsxstrinlinefullplformat}[2]{%
10335   \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}}%
10336   \ifglxstrinsertinside{##2}\fi}%
10337   \ifglxstrinsertinside \else{##2}\fi
10338 }%
10339 }

```

`ong-hyphen-desc` Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```

10340 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
10341 {%
10342   \renewcommand*{\CustomAbbreviationFields}{%
10343     name={\glxstrshortlongdescname},
10344     sort={\glxstrshortlongdescsort},%
10345     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10346     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10347     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10348     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10349 }%

```

```

10350 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10351   \csdef{glxtrpostlink\glscategorylabel}{%
10352     \glxtrifwasfirstuse
10353     {%
10354       \glxtrposthyphenlong{\glslabel}{\glssinsert}%
10355     }%
10356     {%

```

Put the insertion into the post-link:

```

10357       \glxtrposthyphensubsequent{\glslabel}{\glssinsert}%
10358     }%
10359   }%
10360   \glshasattribute{\the\glslabeltok}{regular}%
10361   {%
10362     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10363   }%
10364   {}%
10365 }%
10366 }%
10367 {%
10368   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
10369 }

```

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`

```

10370 \newcommand*{\glabbrvonlyfont}{\glabbrvdefaultfont}%

```

`stabbrvonlyfont`

```

10371 \newcommand*{\glfirstabbrvonlyfont}{\glabbrvonlyfont}%

```

`glslongonlyfont`

```

10372 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%

```

`rstlongonlyfont`

```

10373 \newcommand*{\glfirstlongonlyfont}{\glslongonlyfont}%

```

The default short form suffix:

`lsxtronlysuffix`

```

10374 \newcommand*{\glxtronlysuffix}{\glxtrabbrvpluralsuffix}

```

`\glxtronlyname` The default name format for this style.

```

10375 \newcommand*{\glxtronlyname}{%
10376   \protect\glabbrvonlyfont{\the\glssshorttok}%
10377 }

```

only-short-only

```
10378 \newabbreviationstyle{long-only-short-only}%
10379 {%
10380   \renewcommand*{\CustomAbbreviationFields}{%
10381     name={\glxtronlyname},
10382     sort={\the\glsshorttok},
10383     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10384     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10385     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
10386     description={\protect\glslongonlyfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
10387 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10388   \glshasattribute{\the\glslabeltok}{regular}%
10389   {%
10390     \glissetattribute{\the\glslabeltok}{regular}{false}%
10391   }%
10392   {}%
10393 }%
10394 }%
10395 {%
10396   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtronlysuffix}%
10397   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
10398   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
10399   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
10400   \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%
```

The first use full form doesn't show the short form.

```
10401 \renewcommand*{\glxtrfullformat}[2]{%
10402   \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10403   \ifglxtrinsertinside\else##2\fi
10404 }%
10405 \renewcommand*{\glxtrfullplformat}[2]{%
10406   \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10407   \ifglxtrinsertinside\else##2\fi
10408 }%
10409 \renewcommand*{\Glsxtrfullformat}[2]{%
10410   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10411   \ifglxtrinsertinside\else##2\fi
10412 }%
10413 \renewcommand*{\Glsxtrfullplformat}[2]{%
10414   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10415   \ifglxtrinsertinside\else##2\fi
10416 }%
```

The inline full form does show the short form.

```
10417 \renewcommand*{\glxtrinlinefullformat}[2]{%
10418   \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10419   \ifglxtrinsertinside\else##2\fi
10420   \glxtrfullsep{##1}%
```

```

10421 \glxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshort{##1}}}%
10422 }%
10423 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10424 \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10425 \ifglxtrinsertinside\else##2\fi
10426 \glxtrfullsep{##1}%
10427 \glxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
10428 }%
10429 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10430 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10431 \ifglxtrinsertinside\else##2\fi
10432 \glxtrfullsep{##1}%
10433 \glxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
10434 }%
10435 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10436 \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10437 \ifglxtrinsertinside\else##2\fi
10438 \glxtrfullsep{##1}%
10439 \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10440 }%
10441 }

```

xtronlydescsort

```

10442 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtronlydescname

```

10443 \newcommand*{\glxtronlydescname}{%
10444 \protect\glslongfont{\the\glslongtok}}%
10445 }

```

short-only-desc

```

10446 \newabbreviationstyle{long-only-short-only-desc}%
10447 {%
10448 \renewcommand*{\CustomAbbreviationFields}{%
10449 name={\glxtronlydescname},
10450 sort={\glxtronlydescsort},%
10451 first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
10452 firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
10453 text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
10454 plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}}%
10455 }%

```

Unset the regular attribute if it has been set.

```

10456 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10457 \glshasattribute{\the\glslabeltok}{regular}%
10458 {%
10459 \glissetattribute{\the\glslabeltok}{regular}{false}%
10460 }%
10461 {}%

```

```

10462 }%
10463 }%
10464 {%
10465   \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
10466 }

```

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:

```
10467 \let\@glxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```

10468 \renewcommand*{\markright}[1]{%
10469   \glxtrmarkhook
10470   \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
10471   \glxtrrestoremarkhook
10472 }

```

`\markboth` Save original definition:

```
10473 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```

10474 \renewcommand*{\markboth}[2]{%
10475   \glxtrmarkhook

```

```

10476 \@glsxtr@org@markboth
10477   {\@glsxtrinmark#1\@glsxtrnotinmark}%
10478   {\@glsxtrinmark#2\@glsxtrnotinmark}%
10479 \glsxtrrestoremarkhook
10480 }

```

Also do this for \@starttoc

\@starttoc Save original definition:

```

10481 \let\@glsxtr@org@@starttoc\@starttoc

```

Redefine:

```

10482 \renewcommand*{\@starttoc}[1]{%
10483   \glsxtrmarkhook
10484   \@glsxtrinmark
10485   \@glsxtr@org@@starttoc{#1}%
10486   \@glsxtrnotinmark
10487   \glsxtrrestoremarkhook
10488 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```

10489 \newcommand*{\glsxtrRevertMarks}{%
10490   \let\markright\@glsxtr@org@markright
10491   \let\markboth\@glsxtr@org@markboth
10492   \let\@starttoc\@glsxtr@org@@starttoc
10493 }

```

\glsxtrifinmark

```

10494 \newcommand*{\glsxtrifinmark}[2]{#2}

```

\@glsxtrinmark

```

10495 \newrobustcmd*{\@glsxtrinmark}{%
10496   \let\glsxtrifinmark\@firstoftwo
10497 }

```

glsxtrnotinmark

```

10498 \newrobustcmd*{\@glsxtrnotinmark}{%
10499   \let\glsxtrifinmark\@secondoftwo
10500 }

```

eorpdforheading

```

10501 \ifdef\texorpdfstring
10502 {
10503   \newcommand*{\glsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
10504 }
10505 {
10506   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
10507 }

```

`\glxtrmarkhook` Hook used in new definition of `\markboth` and `\markright` to make some changes to apply to the marks:

```
10508 \newcommand*{\glxtrmarkhook}{%
```

Save current definitions:

```
10509 \let\@glxtr@org@MakeUppercase\MakeUppercase
10510 \let\@glxtr@org@glxtrtitleorpdforheading\glxtrtitleorpdforheading
10511 \let\@glxtr@org@glxtrtitleshort\glxtrtitleshort
10512 \let\@glxtr@org@glxtrtitleshortpl\glxtrtitleshortpl
10513 \let\@glxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
10514 \let\@glxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
10515 \let\@glxtr@org@glxtrtitlename\glxtrtitlename
10516 \let\@glxtr@org@Glsxtrtitlename\Glsxtrtitlename
10517 \let\@glxtr@org@glxtrtitletext\glxtrtitletext
10518 \let\@glxtr@org@Glsxtrtitletext\Glsxtrtitletext
10519 \let\@glxtr@org@glxtrtitleplural\glxtrtitleplural
10520 \let\@glxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
10521 \let\@glxtr@org@glxtrtitlefirst\glxtrtitlefirst
10522 \let\@glxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
10523 \let\@glxtr@org@glxtrtitlefirstplural\glxtrtitlefirstplural
10524 \let\@glxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
10525 \let\@glxtr@org@glxtrtitlelong\glxtrtitlelong
10526 \let\@glxtr@org@glxtrtitlelongpl\glxtrtitlelongpl
10527 \let\@glxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
10528 \let\@glxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
10529 \let\@glxtr@org@glxtrtitlefull\glxtrtitlefull
10530 \let\@glxtr@org@glxtrtitlefullpl\glxtrtitlefullpl
10531 \let\@glxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
10532 \let\@glxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```
10533 \let\glxtrifinmark\@firstoftwo
10534 \let\MakeUppercase\MakeTextUppercase
10535 \let\glxtrtitleorpdforheading\@thirdofthree
10536 \let\glxtrtitleshort\glxtrheadshort
10537 \let\glxtrtitleshortpl\glxtrheadshortpl
10538 \let\Glsxtrtitleshort\Glsxtrheadshort
10539 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
10540 \let\glxtrtitlename\glxtrheadname
10541 \let\Glsxtrtitlename\Glsxtrheadname
10542 \let\glxtrtitletext\glxtrheadtext
10543 \let\Glsxtrtitletext\Glsxtrheadtext
10544 \let\glxtrtitleplural\glxtrheadplural
10545 \let\Glsxtrtitleplural\Glsxtrheadplural
10546 \let\glxtrtitlefirst\glxtrheadfirst
10547 \let\Glsxtrtitlefirst\Glsxtrheadfirst
10548 \let\glxtrtitlefirstplural\glxtrheadfirstplural
10549 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
10550 \let\glxtrtitlelong\glxtrheadlong
10551 \let\glxtrtitlelongpl\glxtrheadlongpl
```

```

10552 \let\Glsxtrtitlelong\Glsxtrheadlong
10553 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
10554 \let\glxstrtitlefull\glxstrheadfull
10555 \let\glxstrtitlefullpl\glxstrheadfullpl
10556 \let\Glsxtrtitlefull\Glsxtrheadfull
10557 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
10558 }

```

`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.)

```

10559 \newcommand*{\glxstrrestoremarkhook}{%
10560 \let\glxstrifinmark\@secondoftwo
10561 \let\MakeUppercase\@glxstr@org@MakeUppercase
10562 \let\glxstrtitleorpdforheading\@glxstr@org@glxstrtitleorpdforheading
10563 \let\glxstrtitleshort\@glxstr@org@glxstrtitleshort
10564 \let\glxstrtitleshortpl\@glxstr@org@glxstrtitleshortpl
10565 \let\Glsxtrtitleshort\@glxstr@org@Glsxtrtitleshort
10566 \let\Glsxtrtitleshortpl\@glxstr@org@Glsxtrtitleshortpl
10567 \let\glxstrtitlename\@glxstr@org@glxstrtitlename
10568 \let\Glsxtrtitlename\@glxstr@org@Glsxtrtitlename
10569 \let\glxstrtitletext\@glxstr@org@glxstrtitletext
10570 \let\Glsxstrtitletext\@glxstr@org@Glsxstrtitletext
10571 \let\glxstrtitleplural\@glxstr@org@glxstrtitleplural
10572 \let\Glsxstrtitleplural\@glxstr@org@Glsxstrtitleplural
10573 \let\glxstrtitlefirst\@glxstr@org@glxstrtitlefirst
10574 \let\Glsxstrtitlefirst\@glxstr@org@Glsxstrtitlefirst
10575 \let\glxstrtitlefirstplural\@glxstr@org@glxstrtitlefirstplural
10576 \let\Glsxstrtitlefirstplural\@glxstr@org@Glsxstrtitlefirstplural
10577 \let\glxstrtitlelong\@glxstr@org@glxstrtitlelong
10578 \let\glxstrtitlelongpl\@glxstr@org@glxstrtitlelongpl
10579 \let\Glsxstrtitlelong\@glxstr@org@Glsxstrtitlelong
10580 \let\Glsxstrtitlelongpl\@glxstr@org@Glsxstrtitlelongpl
10581 \let\glxstrtitlefull\@glxstr@org@glxstrtitlefull
10582 \let\glxstrtitlefullpl\@glxstr@org@glxstrtitlefullpl
10583 \let\Glsxstrtitlefull\@glxstr@org@Glsxstrtitlefull
10584 \let\Glsxstrtitlefullpl\@glxstr@org@Glsxstrtitlefullpl
10585 }

```

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.

```

10586 \newcommand*{\glxstrheadshort}[1]{%
10587 \protect\NoCaseChange
10588 {%
10589 \gl@ifattribute{#1}{headuc}{true}%
10590 {%

```



```

10591 \GLSxtrshort[noindex,hyper=false]{#1}[]%
10592 }%
10593 {%
10594 \glxtrshort[noindex,hyper=false]{#1}[]%
10595 }%
10596 }%
10597 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

10598 \newrobustcmd*{\lsxtrtitleshort}[1]{%
10599 \glxtrshort[noindex,hyper=false]{#1}[]%
10600 }

```

sxtrheadshortpl Command used to display plural short form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrshortpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```

10601 \newcommand*{\glxtrheadshortpl}[1]{%
10602 \protect\NoCaseChange
10603 {%
10604 \gl@ifattribute{#1}{headuc}{true}%
10605 {%
10606 \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
10607 }%
10608 {%
10609 \glxtrshortpl[noindex,hyper=false]{#1}[]%
10610 }%
10611 }%
10612 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents.

```

10613 \newrobustcmd*{\lsxtrtitleshortpl}[1]{%
10614 \glxtrshortpl[noindex,hyper=false]{#1}[]%
10615 }

```

Glsxtrheadshort Command used to display short form in the page header with the first letter converted to upper case.

```

10616 \newcommand*{\Glsxtrheadshort}[1]{%
10617 \protect\NoCaseChange
10618 {%
10619 \gl@ifattribute{#1}{headuc}{true}%
10620 {%
10621 \GLSxtrshort[noindex,hyper=false]{#1}[]%
10622 }%
10623 {%
10624 \Glsxtrshort[noindex,hyper=false]{#1}[]%
10625 }%
10626 }%
10627 }

```

`\lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
10628 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
10629   \Glsxtrshort[noindex,hyper=false]{#1}[]%
10630 }
```

`\sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
10631 \newcommand*{\Glsxtrheadshortpl}[1]{%
10632   \protect\NoCaseChange
10633   {%
10634     \glsifattribute{#1}{headuc}{true}%
10635     {%
10636       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
10637     }%
10638     {%
10639       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
10640     }%
10641   }%
10642 }
```

`\xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
10643 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
10644   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
10645 }
```

`\glsxtrheadname` As above but for the name value.

```
10646 \newcommand*{\glsxtrheadname}[1]{%
10647   \protect\NoCaseChange
10648   {%
10649     \glsifattribute{#1}{headuc}{true}%
10650     {%
10651       \GLSname[noindex,hyper=false]{#1}[]%
10652     }%
10653     {%
10654       \glsname[noindex,hyper=false]{#1}[]%
10655     }%
10656   }%
10657 }
```

`\glsxtrtitlename` Command to display name value in section title and table of contents.

```
10658 \newrobustcmd*{\glsxtrtitlename}[1]{%
10659   \glsname[noindex,hyper=false]{#1}[]%
10660 }
```

`\Glsxtrheadname` First letter converted to upper case

```
10661 \newcommand*{\Glsxtrheadname}[1]{%
```

```

10662 \protect\NoCaseChange
10663 {%
10664   \glsifattribute{#1}{headuc}{true}%
10665   {%
10666     \GLSname[noindex,hyper=false]{#1}[]%
10667   }%
10668   {%
10669     \Glsname[noindex,hyper=false]{#1}[]%
10670   }%
10671 }%
10672 }

```

Glsxtrtitlename Command to display name value in section title and table of contents with the first letter changed to upper case.

```

10673 %\changes{1.21}{2017-11-03}{new}
10674 \newrobustcmd*{\Glsxtrtitlename}[1]{%
10675   \Glsname[noindex,hyper=false]{#1}[]%
10676 }

```

\glsxtrheadtext As above but for the text value.

```

10677 \newcommand*{\glsxtrheadtext}[1]{%
10678   \protect\NoCaseChange
10679   {%
10680     \glsifattribute{#1}{headuc}{true}%
10681     {%
10682       \GLStext[noindex,hyper=false]{#1}[]%
10683     }%
10684     {%
10685       \glstext[noindex,hyper=false]{#1}[]%
10686     }%
10687   }%
10688 }

```

glsxtrtitletext Command to display text value in section title and table of contents.

```

10689 \newrobustcmd*{\glsxtrtitletext}[1]{%
10690   \glstext[noindex,hyper=false]{#1}[]%
10691 }

```

\Glsxtrheadtext First letter converted to upper case

```

10692 \newcommand*{\Glsxtrheadtext}[1]{%
10693   \protect\NoCaseChange
10694   {%
10695     \glsifattribute{#1}{headuc}{true}%
10696     {%
10697       \GLStext[noindex,hyper=false]{#1}[]%
10698     }%
10699     {%
10700       \Glsstext[noindex,hyper=false]{#1}[]%
10701     }%

```

```
10702 }%
10703 }
```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```
10704 \newrobustcmd*{\Glsxtrtitletext}[1]{%
10705   \Gls{text[noindex,hyper=false]}{#1}[]%
10706 }
```

lsxtrheadplural As above but for the plural value.

```
10707 \newcommand*{\Glsxtrheadplural}[1]{%
10708   \protect\NoCaseChange
10709   {%
10710     \glsifattribute{#1}{headuc}{true}%
10711     {%
10712       \GLSplural[noindex,hyper=false]{#1}[]%
10713     }%
10714     {%
10715       \glsplural[noindex,hyper=false]{#1}[]%
10716     }%
10717   }%
10718 }
```

sxtrtitleplural Command to display plural value in section title and table of contents.

```
10719 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
10720   \glsplural[noindex,hyper=false]{#1}[]%
10721 }
```

lsxtrheadplural Convert first letter to upper case.

```
10722 \newcommand*{\Glsxtrheadplural}[1]{%
10723   \protect\NoCaseChange
10724   {%
10725     \glsifattribute{#1}{headuc}{true}%
10726     {%
10727       \GLSplural[noindex,hyper=false]{#1}[]%
10728     }%
10729     {%
10730       \Glsplural[noindex,hyper=false]{#1}[]%
10731     }%
10732   }%
10733 }
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
10734 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
10735   \Glsplural[noindex,hyper=false]{#1}[]%
10736 }
```

glsxtrheadfirst As above but for the first value.

```
10737 \newcommand*{\glsxtrheadfirst}[1]{%
10738   \protect\NoCaseChange
10739   {%
10740     \glsifattribute{#1}{headuc}{true}%
10741     {%
10742       \GLSfirst[noindex,hyper=false]{#1}[]%
10743     }%
10744     {%
10745       \glsfirst[noindex,hyper=false]{#1}[]%
10746     }%
10747   }%
10748 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
10749 \newrobustcmd*{\glsxtrtitlefirst}[1]{%
10750   \glsfirst[noindex,hyper=false]{#1}[]%
10751 }
```

Glsxtrheadfirst First letter converted to upper case

```
10752 \newcommand*{\Glsxtrheadfirst}[1]{%
10753   \protect\NoCaseChange
10754   {%
10755     \glsifattribute{#1}{headuc}{true}%
10756     {%
10757       \GLSfirst[noindex,hyper=false]{#1}[]%
10758     }%
10759     {%
10760       \Glsfirst[noindex,hyper=false]{#1}[]%
10761     }%
10762   }%
10763 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```
10764 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
10765   \Glsfirst[noindex,hyper=false]{#1}[]%
10766 }
```

headfirstplural As above but for the firstplural value.

```
10767 \newcommand*{\glsxtrheadfirstplural}[1]{%
10768   \protect\NoCaseChange
10769   {%
10770     \glsifattribute{#1}{headuc}{true}%
10771     {%
10772       \GLSfirstplural[noindex,hyper=false]{#1}[]%
10773     }%
10774     {%
```

```

10775     \glsfirstplural[noindex,hyper=false]{#1}[]%
10776   }%
10777 }%
10778 }

```

titlefirstplural Command to display firstplural value in section title and table of contents.

```

10779 \newrobustcmd*{\glsxtrtitlefirstplural}[1]{%
10780   \glsfirstplural[noindex,hyper=false]{#1}[]%
10781 }

```

headfirstplural First letter converted to upper case

```

10782 \newcommand*{\Glsxtrheadfirstplural}[1]{%
10783   \protect\NoCaseChange
10784   {%
10785     \glsifattribute{#1}{headuc}{true}%
10786     {%
10787       \GLSfirstplural[noindex,hyper=false]{#1}[]%
10788     }%
10789     {%
10790       \Glsfirstplural[noindex,hyper=false]{#1}[]%
10791     }%
10792   }%
10793 }

```

titlefirstplural Command to display first value in section title and table of contents with the first letter changed to upper case.

```

10794 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
10795   \Glsfirstplural[noindex,hyper=false]{#1}[]%
10796 }

```

\glsxtrheadlong Command used to display long form in the page header.

```

10797 \newcommand*{\glsxtrheadlong}[1]{%
10798   \protect\NoCaseChange
10799   {%
10800     \glsifattribute{#1}{headuc}{true}%
10801     {%
10802       \GLSxtrlong[noindex,hyper=false]{#1}[]%
10803     }%
10804     {%
10805       \glsxtrlong[noindex,hyper=false]{#1}[]%
10806     }%
10807   }%
10808 }

```

glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents.

```

10809 \newrobustcmd*{\glsxtrtitlelong}[1]{%
10810   \glsxtrlong[noindex,hyper=false]{#1}[]%
10811 }

```

`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.

```

10812 \newcommand*{\glxtrheadlongpl}[1]{%
10813   \protect\NoCaseChange
10814   {%
10815     \glsifattribute{#1}{headuc}{true}%
10816     {%
10817       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
10818     }%
10819     {%
10820       \glxtrlongpl[noindex,hyper=false]{#1}[]%
10821     }%
10822   }%
10823 }

```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```

10824 \newrobustcmd*{\glxtrtitlelongpl}[1]{%
10825   \glxtrlongpl[noindex,hyper=false]{#1}[]%
10826 }

```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```

10827 \newcommand*{\Glsxtrheadlong}[1]{%
10828   \protect\NoCaseChange
10829   {%
10830     \glsifattribute{#1}{headuc}{true}%
10831     {%
10832       \GLSxtrlong[noindex,hyper=false]{#1}[]%
10833     }%
10834     {%
10835       \Glsxtrlong[noindex,hyper=false]{#1}[]%
10836     }%
10837   }%
10838 }

```

`Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

10839 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
10840   \Glsxtrlong[noindex,hyper=false]{#1}[]%
10841 }

```

`lsxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```

10842 \newcommand*{\Glsxtrheadlongpl}[1]{%
10843   \protect\NoCaseChange
10844   {%
10845     \glsifattribute{#1}{headuc}{true}%

```

```

10846   {%
10847     \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
10848   }%
10849   {%
10850     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
10851   }%
10852 }%
10853 }

```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

10854 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
10855   \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
10856 }

```

\glsxtrheadfull Command used to display full form in the page header.

```

10857 \newcommand*{\glsxtrheadfull}[1]{%
10858   \protect\NoCaseChange
10859   {%
10860     \glsifattribute{#1}{headuc}{true}%
10861     {%
10862       \GLSxtrfull[noindex,hyper=false]{#1}[]%
10863     }%
10864     {%
10865       \glsxtrfull[noindex,hyper=false]{#1}[]%
10866     }%
10867   }%
10868 }

```

glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents.

```

10869 \newrobustcmd*{\glsxtrtitlefull}[1]{%
10870   \glsxtrfull[noindex,hyper=false]{#1}[]%
10871 }

```

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.

```

10872 \newcommand*{\glsxtrheadfullpl}[1]{%
10873   \protect\NoCaseChange
10874   {%
10875     \glsifattribute{#1}{headuc}{true}%
10876     {%
10877       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
10878     }%
10879     {%
10880       \glsxtrfullpl[noindex,hyper=false]{#1}[]%
10881     }%
10882   }%
10883 }

```


`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```
10884 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
10885   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
10886 }
```

`\Glsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```
10887 \newcommand*{\Glsxtrheadfull}[1]{%
10888   \protect\NoCaseChange
10889   {%
10890     \glsifattribute{#1}{headuc}{true}%
10891     {%
10892       \Glsxtrfull[noindex,hyper=false]{#1}[]%
10893     }%
10894     {%
10895       \Glsxtrfull[noindex,hyper=false]{#1}[]%
10896     }%
10897   }%
10898 }
```

`Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
10899 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
10900   \Glsxtrfull[noindex,hyper=false]{#1}[]%
10901 }
```

`lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```
10902 \newcommand*{\Glsxtrheadfullpl}[1]{%
10903   \protect\NoCaseChange
10904   {%
10905     \glsifattribute{#1}{headuc}{true}%
10906     {%
10907       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
10908     }%
10909     {%
10910       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
10911     }%
10912   }%
10913 }
```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
10914 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
10915   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
10916 }
```

`\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.

```
10917 \ifdef\texorpdfstring
10918 {
10919   \newcommand*\glsfmtshort}[1]{%
10920     \texorpdfstring
10921       {\glsxtrtitleshort{#1}}%
10922       {\glsentryshort{#1}}%
10923   }
10924 }
10925 {
10926   \newcommand*\glsfmtshort}[1]{%
10927     \glsxtrtitleshort{#1}}
10928 }
```

Similarly for the plural version.

`\glsfmtshortpl`

```
10929 \ifdef\texorpdfstring
10930 {
10931   \newcommand*\glsfmtshortpl}[1]{%
10932     \texorpdfstring
10933       {\glsxtrtitleshortpl{#1}}%
10934       {\glsentryshortpl{#1}}%
10935   }
10936 }
10937 {
10938   \newcommand*\glsfmtshortpl}[1]{%
10939     \glsxtrtitleshortpl{#1}}
10940 }
```

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).

```
10941 \ifdef\texorpdfstring
10942 {
10943   \newcommand*\Glsfmtshort}[1]{%
10944     \texorpdfstring
10945       {\Glsxtrtitleshort{#1}}%
10946       {\glsentryshort{#1}}%
10947   }
10948 }
10949 {
10950   \newcommand*\Glsfmtshort}[1]{%
10951     \Glsxtrtitleshort{#1}}
10952 }
```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

10953 \ifdef\texorpdfstring
10954 {
10955   \newcommand*\Glsfmtshortpl}[1]{%
10956     \texorpdfstring
10957     {\Glsxtrtitleshortpl{#1}}%
10958     {\glsentryshortpl{#1}}%
10959   }
10960 }
10961 {
10962   \newcommand*\Glsfmtshortpl}[1]{%
10963     \Glsxtrtitleshortpl{#1}}
10964 }

```

`\glsfmtname` As above but for the name value.

```

10965 \ifdef\texorpdfstring
10966 {
10967   \newcommand*\glsfmtname}[1]{%
10968     \texorpdfstring
10969     {\glsxtrtitlename{#1}}%
10970     {\glsentryname{#1}}%
10971   }
10972 }
10973 {
10974   \newcommand*\glsfmtname}[1]{%
10975     \glsxtrtitlename{#1}}
10976 }

```

`\Glsfmtname` First letter converted to upper case.

```

10977 \ifdef\texorpdfstring
10978 {
10979   \newcommand*\Glsfmtname}[1]{%
10980     \texorpdfstring
10981     {\Glsxtrtitlename{#1}}%
10982     {\glsentryname{#1}}%
10983   }
10984 }
10985 {
10986   \newcommand*\Glsfmtname}[1]{%
10987     \Glsxtrtitlename{#1}}
10988 }

```

`\glsfmttext` As above but for the text value.

```

10989 \ifdef\texorpdfstring
10990 {
10991   \newcommand*\glsfmttext}[1]{%
10992     \texorpdfstring
10993     {\glsxtrtitletext{#1}}%
10994     {\glsentrytext{#1}}%
10995   }

```

```

10996 }
10997 {
10998   \newcommand*{\glsfmttext}[1]{%
10999     \glsxtrtitletext{#1}}
11000 }

```

`\Glsfmttext` First letter converted to upper case.

```

11001 \ifdef\texorpdfstring
11002 {
11003   \newcommand*{\Glsfmttext}[1]{%
11004     \texorpdfstring
11005       {\Glsxtrtitletext{#1}}%
11006       {\glsentrytext{#1}}%
11007   }
11008 }
11009 {
11010   \newcommand*{\Glsfmttext}[1]{%
11011     \Glsxtrtitletext{#1}}
11012 }

```

`\glsfmtplural` As above but for the plural value.

```

11013 \ifdef\texorpdfstring
11014 {
11015   \newcommand*{\glsfmtplural}[1]{%
11016     \texorpdfstring
11017       {\glsxtrtitleplural{#1}}%
11018       {\glsentryplural{#1}}%
11019   }
11020 }
11021 {
11022   \newcommand*{\glsfmtplural}[1]{%
11023     \glsxtrtitleplural{#1}}
11024 }

```

`\Glsfmtplural` First letter converted to upper case.

```

11025 \ifdef\texorpdfstring
11026 {
11027   \newcommand*{\Glsfmtplural}[1]{%
11028     \texorpdfstring
11029       {\Glsxtrtitleplural{#1}}%
11030       {\glsentryplural{#1}}%
11031   }
11032 }
11033 {
11034   \newcommand*{\Glsfmtplural}[1]{%
11035     \Glsxtrtitleplural{#1}}
11036 }

```

`\glsfmtfirst` As above but for the first value.

```

11037 \ifdef\teorpdfstring
11038 {
11039   \newcommand*\glsfmtfirst}[1]{%
11040     \teorpdfstring
11041     {\glstrtitlefirst{#1}}%
11042     {\glstentryfirst{#1}}%
11043   }
11044 }
11045 {
11046   \newcommand*\glsfmtfirst}[1]{%
11047     \glstrtitlefirst{#1}}
11048 }

```

\Glsfmtfirst First letter converted to upper case.

```

11049 \ifdef\teorpdfstring
11050 {
11051   \newcommand*\Glsfmtfirst}[1]{%
11052     \teorpdfstring
11053     {\Glsstrtitlefirst{#1}}%
11054     {\glstentryfirst{#1}}%
11055   }
11056 }
11057 {
11058   \newcommand*\Glsfmtfirst}[1]{%
11059     \Glsstrtitlefirst{#1}}
11060 }

```

\glsfmtfirstpl As above but for the firstplural value.

```

11061 \ifdef\teorpdfstring
11062 {
11063   \newcommand*\glsfmtfirstpl}[1]{%
11064     \teorpdfstring
11065     {\glstrtitlefirstplural{#1}}%
11066     {\glstentryfirstplural{#1}}%
11067   }
11068 }
11069 {
11070   \newcommand*\glsfmtfirstpl}[1]{%
11071     \glstrtitlefirstplural{#1}}
11072 }

```

\Glsfmtfirstpl First letter converted to upper case.

```

11073 \ifdef\teorpdfstring
11074 {
11075   \newcommand*\Glsfmtfirstpl}[1]{%
11076     \teorpdfstring
11077     {\Glsstrtitlefirstplural{#1}}%
11078     {\glstentryfirstplural{#1}}%
11079   }

```

```

11080 }
11081 {
11082   \newcommand*{\Glsfmtfirstpl}[1]{%
11083     \Glsxtrtitlefirstplural{#1}}
11084 }

```

`\glsfmtlong` As above but for the long value.

```

11085 \ifdef\texorpdfstring
11086 {
11087   \newcommand*{\glsfmtlong}[1]{%
11088     \texorpdfstring
11089     {\glsxtrtitlelong{#1}}%
11090     {\glsentrylong{#1}}%
11091   }
11092 }
11093 {
11094   \newcommand*{\glsfmtlong}[1]{%
11095     \glsxtrtitlelong{#1}}
11096 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11097 \ifdef\texorpdfstring
11098 {
11099   \newcommand*{\Glsfmtlong}[1]{%
11100     \texorpdfstring
11101     {\Glsxtrtitlelong{#1}}%
11102     {\glsentrylong{#1}}%
11103   }
11104 }
11105 {
11106   \newcommand*{\Glsfmtlong}[1]{%
11107     \Glsxtrtitlelong{#1}}
11108 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

11109 \ifdef\texorpdfstring
11110 {
11111   \newcommand*{\glsfmtlongpl}[1]{%
11112     \texorpdfstring
11113     {\glsxtrtitlelongpl{#1}}%
11114     {\glsentrylongpl{#1}}%
11115   }
11116 }
11117 {
11118   \newcommand*{\glsfmtlongpl}[1]{%
11119     \glsxtrtitlelongpl{#1}}
11120 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

11121 \ifdef\txorpdfstring
11122 {
11123   \newcommand*\Glsfmtlongpl[1]{%
11124     \txorpdfstring
11125     {\Glsxtrtitlelongpl{#1}}%
11126     {\glsentrylongpl{#1}}%
11127   }
11128 }
11129 {
11130   \newcommand*\Glsfmtlongpl[1]{%
11131     \Glsxtrtitlelongpl{#1}}
11132 }

```

`\glsfmtfull` In-line full format.

```

11133 \ifdef\txorpdfstring
11134 {
11135   \newcommand*\glsfmtfull[1]{%
11136     \txorpdfstring
11137     {\glsxtrtitlefull{#1}}%
11138     {\glsxtrinlinefullformat{#1}{}}%
11139   }
11140 }
11141 {
11142   \newcommand*\glsfmtfull[1]{%
11143     \glsxtrtitlefull{#1}}
11144 }

```

`\Glsfmtfull` First letter converted to upper case.

```

11145 \ifdef\txorpdfstring
11146 {
11147   \newcommand*\Glsfmtfull[1]{%
11148     \txorpdfstring
11149     {\Glsxtrtitlefull{#1}}%
11150     {\Glsxtrinlinefullformat{#1}{}}%
11151   }
11152 }
11153 {
11154   \newcommand*\Glsfmtfull[1]{%
11155     \Glsxtrtitlefull{#1}}
11156 }

```

`\glsfmtfullpl` In-line full plural format.

```

11157 \ifdef\txorpdfstring
11158 {
11159   \newcommand*\glsfmtfullpl[1]{%
11160     \txorpdfstring
11161     {\glsxtrtitlefullpl{#1}}%
11162     {\glsxtrinlinefullplformat{#1}{}}%
11163   }

```

```

11164 }
11165 {
11166   \newcommand*{\Glsfmtfullpl}[1]{%
11167     \Glsxtrtitlefullpl{#1}}
11168 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

11169 \ifdef\texorpdfstring
11170 {
11171   \newcommand*{\Glsfmtfullpl}[1]{%
11172     \texorpdfstring
11173     {\Glsxtrtitlefullpl{#1}}%
11174     {\Glsxtrinlinefullplformat{#1}{}}%
11175   }
11176 }
11177 {
11178   \newcommand*{\Glsfmtfullpl}[1]{%
11179     \Glsxtrtitlefullpl{#1}}
11180 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

11181 \newcommand*{\RequireGlossariesExtraLang}[1]{%
11182   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11183 }

```

`sariesExtraLang`

```

11184 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11185   \ProvidesFile{glossariesxtr-#1.ldf}%
11186 }

```

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.

```

11187 \newcommand{\Glsxtr@loaddialect}{%
11188   \IfTrackedLanguageFileExists{\this@dialect}%
11189   {glossariesxtr-}% prefix
11190   {.ldf}%
11191   {%
11192     \RequireGlossariesExtraLang{\CurrentTrackedTag}%

```



```

11193 }%
11194 {}% not found

If glossaries-extra-bib2gls has been loaded, \@glsxtrdialecthook will check for the associ-
ated script, otherwise it will do nothing.

11195 \@glsxtrdialecthook
11196 }

11197 \@ifpackageloaded{tracklang}
11198 {%
11199   \AnyTrackedLanguages
11200   {%
11201     \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
11202   }%
11203   {}%
11204 }
11205 {}

Load glossaries-extra-stylemods if required.
11206 \@glsxtr@redefstyles
and set the style:
11207 \@glsxtr@do@style

```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for bib2gls and is automatically loaded by the record option.

```

11208 \NeedsTeXFormat{LaTeX2e}
11209 \ProvidesPackage{glossaries-extra-bib2gls}[2018/02/26 v1.27 (NLCT)]

```

These are some convenient macros for use with custom rules.

`\glshex`

```

11210 \newcommand*{\glshex}{\string\u}

```

`\rprovidecommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```

11211 \newcommand*{\glsxtrprovidecommand}{\providecommand}

```

Provide missing Greek letters for use in maths mode. These 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 Greek letters. These commands use an upright font for capitals and italic for lower case to provide a better match with the other Greek symbols provided by the kernel.

`\Alpha`

```

11212 \providecommand{\Alpha}{\mathrm{A}}

```

```

\Beta
11213 \providecommand*\Beta{\mathrm{B}}

\Epsilon
11214 \providecommand*\Epsilon{\mathrm{E}}

\Zeta
11215 \providecommand*\Zeta{\mathrm{Z}}

\Eta
11216 \providecommand*\Eta{\mathrm{H}}

\Iota
11217 \providecommand*\Iota{\mathrm{I}}

\Kappa
11218 \providecommand*\Kappa{\mathrm{K}}

\Mu
11219 \providecommand*\Mu{\mathrm{M}}

\Nu
11220 \providecommand*\Nu{\mathrm{N}}

\Omicron
11221 \providecommand*\Omicron{\mathrm{O}}

\Rho
11222 \providecommand*\Rho{\mathrm{P}}

\Tau
11223 \providecommand*\Tau{\mathrm{T}}

\Chi
11224 \providecommand*\Chi{\mathrm{X}}

\Digamma
11225 \providecommand*\Digamma{\mathrm{F}}

\omicron
11226 \providecommand*\omicron{\mathit{o}}

    Provide corresponding upright characters if upgreek has been loaded.
11227 \@ifpackageloaded{upgreek}%
11228 {

```

```

\Upalpha
11229 \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
11230 \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
11231 \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
11232 \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
11233 \providecommand*\Upeta{\mathrm{H}}

\Upiota
11234 \providecommand*\Upiota{\mathrm{I}}

\Upkappa
11235 \providecommand*\Upkappa{\mathrm{K}}

\Upmu
11236 \providecommand*\Upmu{\mathrm{M}}

\Upnu
11237 \providecommand*\Upnu{\mathrm{N}}

\Upomicron
11238 \providecommand*\Upomicron{\mathrm{O}}

\Uprho
11239 \providecommand*\Uprho{\mathrm{P}}

\Uptau
11240 \providecommand*\Uptau{\mathrm{T}}

\Upchi
11241 \providecommand*\Upchi{\mathrm{X}}

\upomicron
11242 \providecommand*\upomicron{\mathrm{o}}

11243 }%
11244 {}% upgreek.sty not loaded

```

This package provides some basic rules but is 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. The file will then be automatically loaded by `glossaries-extra` if the document has support for that language.

When combining these blocks of rules, remember to separate them with the appropriate character. For example:

```
sort-rule={\glxtrcontrolrules
; \glxtrspacerules
; \glxtrnonprintablerules
; \glxtrcombiningdiacriticrules
, \glxtrhyphenrules
< \glxtrgeneralpuncrules
< \glxtrdigitrules
< \glxtrfractionrules
< \glxtrGeneralLatinIVrules
< \glxtrMathItalicGreekIrules
}
```

`xtrcontrolrules` These are control characters that are usually placed at the start of a rule in the ‘ignored characters’ section. `\string` is used for punctuation characters in case they’ve been made active.

```
11245 \newcommand*{\glxtrcontrolrules}{%
11246 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
11247 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
11248 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
11249 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
11250 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
11251 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
11252 0010\string'\string=\glshex 0011
11253 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
11254 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
11255 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
11256 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
11257 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
11258 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
11259 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
11260 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
11261 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
11262 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
11263 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
11264 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
11265 }
```

lsxtrspacerules These are space characters.

```
11266 \newcommand*{\glxtrspacerules}{%
11267 \string' \string'\string;
11268 \string'\glshex 00A0\string'\string;
11269 \string'\glshex 2000\string'\string;
11270 \string'\glshex 2001\string'\string;
11271 \string'\glshex 2002\string'\string;
11272 \string'\glshex 2003\string'\string;
11273 \string'\glshex 2004\string'\string;
11274 \string'\glshex 2005\string'\string;
11275 \string'\glshex 2006\string'\string;
11276 \string'\glshex 2007\string'\string;
11277 \string'\glshex 2008\string'\string;
11278 \string'\glshex 2009\string'\string;
11279 \string'\glshex 200A\string'\string;
11280 \string'\glshex 3000\string'
11281 }
```

nprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```
11282 \newcommand*{\glxtrnonprintablerules}{%
11283 \string'\glshex FEFF\string'\string;
11284 \string'\glshex 000A\string'\string;
11285 \string'\glshex 0009\string'\string;
11286 \string'\glshex 000C\string'\string;
11287 \string'\glshex 000B\string'
11288 }
```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```
11289 \newcommand*{\glxtrcombiningdiacriticrules}{%
11290 \glxtrcombiningdiacriticIrules\string;
11291 \glxtrcombiningdiacriticIIrules\string;
11292 \glxtrcombiningdiacriticIIIrules\string;
11293 \glxtrcombiningdiacriticIVrules
11294 }
```

diacriticIrules First set of combining diacritic marks.

```
11295 \newcommand*{\glxtrcombiningdiacriticIrules}{%
11296 \glshex 0301\string;% combining acute
11297 \glshex 0300\string;% combining grave
11298 \glshex 0306\string;% combining breve
11299 \glshex 0302\string;% combining circumflex
11300 \glshex 030C\string;% combining caron
11301 \glshex 030A\string;% combining ring
11302 \glshex 030D\string;% combining vertical line above
11303 \glshex 0308\string;% combining diaeresis
11304 \glshex 030B\string;% combining double acute
11305 \glshex 0303\string;% combining tilde
11306 \glshex 0307\string;% combining dot above
11307 \glshex 0304% combining macron
```

11308 }

iacriticIIrules Second set of combining diacritic marks.

```
11309 \newcommand*{\glxtrcombingdiacriticIIrules}{%
11310 \glshex 0337\string;% combining short solidus overlay
11311 \glshex 0327\string;% combining cedilla
11312 \glshex 0328\string;% combining ogonek
11313 \glshex 0323\string;% combining dot below
11314 \glshex 0332\string;% combining low line
11315 \glshex 0305\string;% combining overline
11316 \glshex 0309\string;% combining hook above
11317 \glshex 030E\string;% combining double vertical line above
11318 \glshex 030F\string;% combining double grave accent
11319 \glshex 0310\string;% combining candrabindu
11320 \glshex 0311\string;% combining inverted breve
11321 \glshex 0312\string;% combining turned comma above
11322 \glshex 0313\string;% combining comma above
11323 \glshex 0314\string;% combining reversed comma above
11324 \glshex 0315\string;% combining comma above right
11325 \glshex 0316\string;% combining grave accent below
11326 \glshex 0317% combining acute accent below
11327 }
```

acriticIIIrules Third set of combining diacritic marks.

```
11328 \newcommand*{\glxtrcombingdiacriticIIIrules}{%
11329 \glshex 0318\string;% combining left tack below
11330 \glshex 0319\string;% combining right tack below
11331 \glshex 031A\string;% combining left angle above
11332 \glshex 031B\string;% combining horn
11333 \glshex 031C\string;% combining left half ring below
11334 \glshex 031D\string;% combining up tack below
11335 \glshex 031E\string;% combining down tack below
11336 \glshex 031F\string;% combining plus sign below
11337 \glshex 0320\string;% combining minus sign below
11338 \glshex 0321\string;% combining palatalized hook below
11339 \glshex 0322\string;% combining retroflex hook below
11340 \glshex 0324\string;% combining diaeresis below
11341 \glshex 0325\string;% combining ring below
11342 \glshex 0326\string;% combining comma below
11343 \glshex 0329\string;% combining vertical line below
11344 \glshex 032A\string;% combining bridge below
11345 \glshex 032B\string;% combining inverted double arch below
11346 \glshex 032C\string;% combining caron below
11347 \glshex 032D\string;% combining circumflex accent below
11348 \glshex 032E\string;% combining breve below
11349 \glshex 032F\string;% combining inverted breve below
11350 \glshex 0330\string;% combining tilde below
11351 \glshex 0331\string;% combining macron below
11352 \glshex 0333\string;% combining double low line
```

```

11353 \glshex 0334\string;% combining tilde overlay
11354 \glshex 0335\string;% combining short stroke overlay
11355 \glshex 0336\string;% combining long stroke overlay
11356 \glshex 0338\string;% combining long solidus overlay
11357 \glshex 0339\string;% combining combining right half ring below
11358 \glshex 033A\string;% combining inverted bridge below
11359 \glshex 033B\string;% combining square below
11360 \glshex 033C\string;% combining seagull below
11361 \glshex 033D\string;% combining x above
11362 \glshex 033E\string;% combining vertical tilde
11363 \glshex 033F\string;% combining double overline
11364 \glshex 0342\string;% combining Greek perispomeni
11365 \glshex 0344\string;% combining Greek dialytika tonos
11366 \glshex 0345\string;% combining Greek ypogegrammeni
11367 \glshex 0360\string;% combining double tilde
11368 \glshex 0361\string;% combining double inverted breve
11369 \glshex 0483\string;% combining Cyrillic titlo
11370 \glshex 0484\string;% combining Cyrillic palatalization
11371 \glshex 0485\string;% combining Cyrillic dasia pneumata
11372 \glshex 0486% combining Cyrillic psili pneumata
11373 }

```

iacriticIVrules Fourth set of combining diacritic marks.

```

11374 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
11375 \glshex 20D0\string;% combining left harpoon above
11376 \glshex 20D1\string;% combining right harpoon above
11377 \glshex 20D2\string;% combining long vertical line overlay
11378 \glshex 20D3\string;% combining short vertical line overlay
11379 \glshex 20D4\string;% combining anticlockwise arrow above
11380 \glshex 20D5\string;% combining clockwise arrow above
11381 \glshex 20D6\string;% combining left arrow above
11382 \glshex 20D7\string;% combining right arrow above
11383 \glshex 20D8\string;% combining ring overlay
11384 \glshex 20D9\string;% combining clockwise ring overlay
11385 \glshex 20DA\string;% combining anticlockwise ring overlay
11386 \glshex 20DB\string;% combining three dots above
11387 \glshex 20DC\string;% combining four dots above
11388 \glshex 20DD\string;% combining enclosing circle
11389 \glshex 20DE\string;% combining enclosing square
11390 \glshex 20DF\string;% combining enclosing diamond
11391 \glshex 20E0\string;% combining enclosing circle backslash
11392 \glshex 20E1% combining left right arrow above
11393 }

```

sxtrhyphenrules Hyphens.

```

11394 \newcommand*{\glxtrhyphenrules}{%
11395 \string'\string-\string'\string;% ASCII hyphen
11396 \glshex 00AD\string;% soft hyphen
11397 \glshex 2010\string;% hyphen

```

```

11398 \glshex 2011\string;% non-breaking hyphen
11399 \glshex 2012\string;% figure dash
11400 \glshex 2013\string;% en dash
11401 \glshex 2014\string;% em dash
11402 \glshex 2015\string;% horizontal bar
11403 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
11404 }

```

eneralpuncrules General punctuation.

```

11405 \newcommand*{\glxtrgeneralpuncrules}{%
11406 \glxtrgeneralpuncIrules
11407 \string<\glxtrcurrencyrules
11408 \string<\glxtrgeneralpuncIIrules
11409 }

```

eneralpuncIrules First set of general punctuation.

```

11410 \newcommand*{\glxtrgeneralpuncIrules}{%
11411 \string'\glshex 005F\string'% underscore
11412 \string<\glshex 00AF% macron
11413 \string<\string'\glshex 002C\string'% comma
11414 \string<\string'\glshex 003B\string'% semi-colon
11415 \string<\string'\glshex 003A\string'% colon
11416 \string<\string'\glshex 0021\string'% exclamation mark
11417 \string<\glshex 00A1% inverted exclamation mark
11418 \string<\string'\glshex 003F\string'% question mark
11419 \string<\glshex 00BF% inverted question mark
11420 \string<\string'\glshex 002F\string'% solidus
11421 \string<\string'\glshex 002E\string'% full stop
11422 \string<\glshex 00B4% acute accent
11423 \string<\string'\glshex 0060\string'% grave accent
11424 \string<\string'\glshex 005E\string'% circumflex accent
11425 \string<\glshex 00A8% diaersis
11426 \string<\string'\glshex 007E\string'% tilde
11427 \string<\glshex 00B7% middle dot
11428 \string<\glshex 00B8% cedilla
11429 \string<\string'\glshex 0027\string'% straight apostrophe
11430 \string<\string'\glshex 0022\string'% straight double quote
11431 \string<\glshex 00AB% left guillemet
11432 \string<\glshex 00BB% right guillemet
11433 \string<\string'\glshex 0028\string'% left parenthesis
11434 \string=\glshex 207D\string=\glshex 208D% super/subscript left parenthesis
11435 \string<\string'\glshex 0029\string'% right parenthesis
11436 \string=\glshex 207E\string=\glshex 208E% super/subscript right parenthesis
11437 \string<\string'\glshex 005B\string'% left square bracket
11438 \string<\string'\glshex 005D\string'% right square bracket
11439 \string<\string'\glshex 007B\string'% left curly bracket
11440 \string<\string'\glshex 007D\string'% right curly bracket
11441 \string<\glshex 00A7% section sign
11442 \string<\glshex 00B6% pilcrow sign

```



```

11443 \string<\glshex 00A9% copyright sign
11444 \string<\glshex 00AE% registered sign
11445 \string<\string'\glshex 0040\string'% at sign
11446 }

```

trcurrencyrules General punctuation.

```

11447 \newcommand*{\glxtrcurrencyrules}{%
11448 \glshex 00A4% currency sign
11449 \string<\glshex 0E3F% Thai currency symbol baht
11450 \string<\glshex 00A2% cent sign
11451 \string<\glshex 20A1% colon sign
11452 \string<\glshex 20A2% cruzeiro sign
11453 \string<\string'\glshex 0024\string'% dollar sign
11454 \string<\glshex 20AB% dong sign
11455 \string<\glshex 20AC% euro sign
11456 \string<\glshex 20A3% French franc sign
11457 \string<\glshex 20A4% lira sign
11458 \string<\glshex 20A5% mill sign
11459 \string<\glshex 20A6% naira sign
11460 \string<\glshex 20A7% peseta sign
11461 \string<\glshex 00A3% pound sign
11462 \string<\glshex 20A8% rupee sign
11463 \string<\glshex 20AA% new sheqel sign
11464 \string<\glshex 20A9% won sign
11465 \string<\glshex 00A5% yen sign
11466 }

```

eralpuncIIrules Second set of general punctuation.

```

11467 \newcommand*{\glxtrgeneralpuncIIrules}{%
11468 \string'\glshex 002A\string'% asterisk
11469 \string<\string'\glshex 005C\string'% backslash
11470 \string<\string'\glshex 0026\string'% ampersand
11471 \string<\string'\glshex 0023\string'% hash sign
11472 \string<\string'\glshex 0025\string'% percent sign
11473 \string<\string'\glshex 002B\string'% plus sign
11474 \string=>\glshex 207A\string=>\glshex 208A% super/subscript plus sign
11475 \string<\glshex 00B1% plus-minus sign
11476 \string<\glshex 00F7% division sign
11477 \string<\glshex 00D7% multiplication sign
11478 \string<\string'\glshex 003C\string'% less-than sign
11479 \string<\string'\glshex 003D\string'% equals sign
11480 \string<\string'\glshex 003E\string'% greater-than sign
11481 \string<\glshex 00AC% not sign
11482 \string<\string'\glshex 007C\string'% vertical bar (pipe)
11483 \string<\glshex 00A6% broken bar
11484 \string<\glshex 00B0% degree sign
11485 \string<\glshex 00B5% micron sign
11486 }

```

eralLatinIrules Basic Latin alphabet.

```
11487 \newcommand*{\glxtrGeneralLatinIrules}{%
11488 \glxtrLatinA
11489 \string<b,B%
11490 \string<c,C%
11491 \string<d,D%
11492 \string<\glxtrLatinE
11493 \string<f,F%
11494 \string<g,G%
11495 \string<\glxtrLatinH
11496 \string<\glxtrLatinI
11497 \string<j,J%
11498 \string<\glxtrLatinK
11499 \string<\glxtrLatinL
11500 \string<\glxtrLatinM
11501 \string<\glxtrLatinN
11502 \string<\glxtrLatinO
11503 \string<\glxtrLatinP
11504 \string<q,Q%
11505 \string<r,R%
11506 \string<\glxtrLatinS
11507 \string<\glxtrLatinT
11508 \string<u,U%
11509 \string<v,V%
11510 \string<w,W%
11511 \string<\glxtrLatinX
11512 \string<y,Y%
11513 \string<z,Z
11514 }
```

eralLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```
11515 \newcommand*{\glxtrGeneralLatinIIrules}{%
11516 \glxtrLatinA
11517 \string<b,B%
11518 \string<c,C%
11519 \string<d,D%
11520 \string<\glxtrLatinEth
11521 \string<\glxtrLatinE
11522 \string<f,F%
11523 \string<g,G%
11524 \string<\glxtrLatinH
11525 \string<\glxtrLatinI
11526 \string<j,J%
11527 \string<\glxtrLatinK
11528 \string<\glxtrLatinL
11529 \string<\glxtrLatinM
11530 \string<\glxtrLatinN
11531 \string<\glxtrLatinO
11532 \string<\glxtrLatinP
```

```

11533 \string<q,Q%
11534 \string<r,R%
11535 \string<\glxtrLatinS
11536 \string& SS \string, \glxtrLatinEszettSs
11537 \string<\glxtrLatinT
11538 \string<u,U%
11539 \string<v,V%
11540 \string<w,W%
11541 \string<\glxtrLatinX
11542 \string<y,Y%
11543 \string<z,Z%
11544 }

```

alLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

11545 \newcommand*{\glxtrGeneralLatinIIIrules}{%
11546 \glxtrLatinA
11547 \string<b,B%
11548 \string<c,C%
11549 \string<d,D%
11550 \string<\glxtrLatinEth
11551 \string<\glxtrLatinE
11552 \string<f,F%
11553 \string<g,G%
11554 \string<\glxtrLatinH
11555 \string<\glxtrLatinI
11556 \string<j,J%
11557 \string<\glxtrLatinK
11558 \string<\glxtrLatinL
11559 \string<\glxtrLatinM
11560 \string<\glxtrLatinN
11561 \string<\glxtrLatinO
11562 \string<\glxtrLatinP
11563 \string<q,Q%
11564 \string<r,R%
11565 \string<\glxtrLatinS
11566 \string& SZ, \glxtrLatinEszettSz
11567 \string<\glxtrLatinT
11568 \string<u,U%
11569 \string<v,V%
11570 \string<w,W%
11571 \string<\glxtrLatinX
11572 \string<y,Y%
11573 \string<z,Z%
11574 }

```

ralLatinIVrules General Latin alphabet (Æ treated as AE and Æ treated as OE, Þ treated as TH, ß treated as SS, eth between D and E).

```

11575 \newcommand*{\glxtrGeneralLatinIVrules}{%
11576 \glxtrLatinA

```

```

11577 \string& AE , \glxtrLatinAELigature
11578 \string<b,B%
11579 \string<c,C%
11580 \string<d,D%
11581 \string<\glxtrLatinEth
11582 \string<\glxtrLatinE
11583 \string<f,F%
11584 \string<g,G%
11585 \string<\glxtrLatinH
11586 \string<\glxtrLatinI
11587 \string<j,J%
11588 \string<\glxtrLatinK
11589 \string<\glxtrLatinL
11590 \string<\glxtrLatinM
11591 \string<\glxtrLatinN
11592 \string<\glxtrLatinO
11593 \string& OE , \glxtrLatinOELigature
11594 \string<\glxtrLatinP
11595 \string<q,Q%
11596 \string<r,R%
11597 \string<\glxtrLatinS
11598 \string& SS , \glxtrLatinEszettSs
11599 \string<\glxtrLatinT
11600 \string& th =\glshex 00DE
11601 \string& TH =\glshex 00FE
11602 \string<u,U%
11603 \string<v,V%
11604 \string<w,W%
11605 \string<\glxtrLatinX
11606 \string<y,Y%
11607 \string<z,Z%
11608 }

```

eralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

11609 \newcommand*{\glxtrGeneralLatinVrules}{%
11610 \glxtrLatinA
11611 \string<b,B%
11612 \string<c,C%
11613 \string<d,D%
11614 \string<\glxtrLatinEth
11615 \string<\glxtrLatinE
11616 \string<f,F%
11617 \string<g,G%
11618 \string<\glxtrLatinH
11619 \string<\glxtrLatinI
11620 \string<j,J%
11621 \string<\glxtrLatinK
11622 \string<\glxtrLatinL
11623 \string<\glxtrLatinM

```

```

11624 \string<\glxtrLatinN
11625 \string<\glxtrLatinO
11626 \string<\glxtrLatinP
11627 \string<q,Q%
11628 \string<r,R%
11629 \string<\glxtrLatinS
11630 \string& SS , \glxtrLatinEszettSs
11631 \string<\glxtrLatinT
11632 \string& th =\glshex 00DE
11633 \string& TH =\glshex 00FE
11634 \string<u,U%
11635 \string<v,V%
11636 \string<w,W%
11637 \string<\glxtrLatinX
11638 \string<y,Y%
11639 \string<z,Z%
11640 }

```

raLatinVIrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

11641 \newcommand*{\glxtrGeneralLatinVIrules}{%
11642 \glxtrLatinA
11643 \string<b,B%
11644 \string<c,C%
11645 \string<d,D%
11646 \string<\glxtrLatinEth
11647 \string<\glxtrLatinE
11648 \string<f,F%
11649 \string<g,G%
11650 \string<\glxtrLatinH
11651 \string<\glxtrLatinI
11652 \string<j,J%
11653 \string<\glxtrLatinK
11654 \string<\glxtrLatinL
11655 \string<\glxtrLatinM
11656 \string<\glxtrLatinN
11657 \string<\glxtrLatinO
11658 \string<\glxtrLatinP
11659 \string<q,Q%
11660 \string<r,R%
11661 \string<\glxtrLatinS
11662 \string& SZ , \glxtrLatinEszettSz
11663 \string<\glxtrLatinT
11664 \string& th =\glshex 00DE
11665 \string& TH =\glshex 00FE
11666 \string<u,U%
11667 \string<v,V%
11668 \string<w,W%
11669 \string<\glxtrLatinX
11670 \string<y,Y%

```

```

11671 \string<z,Z%
11672 }

```

alLatinVIIrules General Latin alphabet (Æ between A and B, eth between D and E, insular G as G, CE between O and P, long S equivalent to S, Þ between T and U and wynn as W).

```

11673 \newcommand*{\glxtrGeneralLatinVIIrules}{%
11674 \glxtrLatinA
11675 \string<\glxtrLatinAELigature
11676 \string<b,B%
11677 \string<c,C%
11678 \string<d,D%
11679 \string<\glxtrLatinEth
11680 \string<\glxtrLatinE
11681 \string<f,F%
11682 \string<\glxtrLatinInsularG
11683 \string<\glxtrLatinH
11684 \string<\glxtrLatinI
11685 \string<j,J%
11686 \string<\glxtrLatinK
11687 \string<\glxtrLatinL
11688 \string<\glxtrLatinM
11689 \string<\glxtrLatinN
11690 \string<\glxtrLatinO
11691 \string<\glxtrLatinOELigature
11692 \string<\glxtrLatinP
11693 \string<q,Q%
11694 \string<r,R%
11695 \string<\glshex 017F=\glxtrLatinS % s and long s
11696 \string<\glxtrLatinT
11697 \string<\glxtrLatinThorn
11698 \string<u,U%
11699 \string<v,V%
11700 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
11701 \string<\glxtrLatinX
11702 \string<y,Y%
11703 \string<z,Z%
11704 }

```

lLatinVIIIrules General Latin alphabet (Æ treated as AE and Ć treated as OE, Þ treated as TH, ß treated as SS, eth treated as D, Ø treated as O, Ł treated as L).

```

11705 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
11706 \glxtrLatinA
11707 \string& AE , \glxtrLatinAELigature
11708 \string<b,B%
11709 \string<c,C%
11710 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
11711 \string<\glxtrLatinE
11712 \string<f,F%
11713 \string<g,G%

```

```

11714 \string<\glxtrLatinH
11715 \string<\glxtrLatinI
11716 \string<j,J%
11717 \string<\glxtrLatinK
11718 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
11719 \string<\glxtrLatinM
11720 \string<\glxtrLatinN
11721 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
11722 \string& OE , \glxtrLatinOELigature
11723 \string<\glxtrLatinP
11724 \string<q,Q%
11725 \string<r,R%
11726 \string<\glxtrLatinS
11727 \string& SS , \glxtrLatinEszettSs
11728 \string<\glxtrLatinT
11729 \string& th =\glshex 00DE
11730 \string& TH =\glshex 00FE
11731 \string<u,U%
11732 \string<v,V%
11733 \string<w,W%
11734 \string<\glxtrLatinX
11735 \string<y,Y%
11736 \string<z,Z%
11737 }

```

\glxtrLatinA

```

11738 \newcommand*{\glxtrLatinA}{%
11739   a\string=\glshex 00AA\string=\glshex 2090,A
11740 }

```

\glxtrLatinE

```

11741 \newcommand*{\glxtrLatinE}{%
11742   e\string=\glshex 2091,E
11743 }

```

\glxtrLatinH

```

11744 \newcommand*{\glxtrLatinH}{%
11745   h\string=\glshex 2095,H
11746 }

```

\glxtrLatinI

```

11747 \newcommand*{\glxtrLatinI}{%
11748   i\string=\glshex 2071,I
11749 }

```

\glxtrLatinK

```

11750 \newcommand*{\glxtrLatinK}{%
11751   k\string=\glshex 2096,K
11752 }

```

\glsxtrLatinL

```
11753 \newcommand*{\glsxtrLatinL}{%
11754   l\string=\glshex 2097,L
11755 }
```

\glsxtrLatinM

```
11756 \newcommand*{\glsxtrLatinM}{%
11757   m\string=\glshex 2098,M
11758 }
```

\glsxtrLatinN

```
11759 \newcommand*{\glsxtrLatinN}{%
11760   n\string=\glshex 207F\string=\glshex 2099,N
11761 }
```

\glsxtrLatinO

```
11762 \newcommand*{\glsxtrLatinO}{%
11763   o\string=\glshex 00BA\string=\glshex 2092,O
11764 }
```

\glsxtrLatinP

```
11765 \newcommand*{\glsxtrLatinP}{%
11766   p\string=\glshex 209A,P
11767 }
```

\glsxtrLatinS

```
11768 \newcommand*{\glsxtrLatinS}{%
11769   s\string=\glshex 209B,S
11770 }
```

\glsxtrLatinT

```
11771 \newcommand*{\glsxtrLatinT}{%
11772   t\string=\glshex 209C,T
11773 }
```

\glsxtrLatinX

```
11774 \newcommand*{\glsxtrLatinX}{%
11775   x\string=\glshex 2093,X
11776 }
```

lsxtrLatinSchwa Latin schwa (lower case, subscript and upper case).

```
11777 \newcommand*{\glsxtrLatinSchwa}{%
11778   \glshex 0259\string=\glshex 2094,\glshex 018F
11779 }
```

trLatinEszettSs

```
11780 \newcommand*{\glsxtrLatinEszettSs}{%
11781   \glshex 00DF% eszett
11782   \string=\glshex 017Fs % long S s
11783 }
```


trLatinEszettSz

```
11784 \newcommand*{\glxtrLatinEszettSz}{%
11785   \glshex 00DF% eszett
11786   \string= \glshex 017Fz % long S z
11787 }
```

\glxtrLatinEth

```
11788 \newcommand*{\glxtrLatinEth}{%
11789   \glshex 00F0,\glshex 00D0% eth
11790 }
```

lsxtrLatinThorn

```
11791 \newcommand*{\glxtrLatinThorn}{%
11792   \glshex 00FE,\glshex 00DE% thorn
11793 }
```

LatinAELigature

```
11794 \newcommand*{\glxtrLatinAELigature}{%
11795   \glshex 00E6,\glshex 00C6% AE-ligature
11796 }
```

LatinOELigature

```
11797 \newcommand*{\glxtrLatinOELigature}{%
11798   \glshex 0153,\glshex 0152% OE-ligature
11799 }
```

\glxtrLatinAA

```
11800 \newcommand*{\glxtrLatinAA}{%
11801   \glshex 00E5=a\glshex 030A,% \aa
11802   \glshex 00C5=A\glshex 030A% \AA
11803 }
```

glxtrLatinWynn

```
11804 \newcommand*{\glxtrLatinWynn}{%
11805   \glshex 01BF,\glshex 01F7% wynn
11806 }
```

trLatinInsularG

```
11807 \newcommand*{\glxtrLatinInsularG}{%
11808   \glshex 1D79,\glshex A77D% insular G
11809   \string; g, G
11810 }
```

sxtrLatinOslash

```
11811 \newcommand*{\glxtrLatinOslash}{%
11812   \glshex 00F8,\glshex 00D8% \o, \O
11813 }
```

sxtrLatinLslash

```
11814 \newcommand*{\glsxtrLatinLslash}{%
11815   \glshex 0142,\glshex 0141% \l, \L
11816 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
11817 \newcommand*{\glsxtrMathUpGreekIrules}{%
11818   \glsxtrUpAlpha
11819   \string<\glsxtrUpBeta
11820   \string<\glsxtrUpGamma
11821   \string<\glsxtrUpDelta
11822   \string<\glsxtrUpEpsilon
11823   \string<\glsxtrUpDigamma
11824   \string<\glsxtrUpZeta
11825   \string<\glsxtrUpEta
11826   \string<\glsxtrUpTheta
11827   \string<\glsxtrUpIota
11828   \string<\glsxtrUpKappa
11829   \string<\glsxtrUpLambda
11830   \string<\glsxtrUpMu
11831   \string<\glsxtrUpNu
11832   \string<\glsxtrUpXi
11833   \string<\glsxtrUpOmicron
11834   \string<\glsxtrUpPi
11835   \string<\glsxtrUpRho
11836   \string<\glsxtrUpSigma
11837   \string<\glsxtrUpTau
11838   \string<\glsxtrUpUpsilon
11839   \string<\glsxtrUpPhi
11840   \string<\glsxtrUpChi
11841   \string<\glsxtrUpPsi
11842   \string<\glsxtrUpOmega
11843 }
```

hUpGreekIIrules Doesn't include digamma.

```
11844 \newcommand*{\glsxtrMathUpGreekIIrules}{%
11845   \glsxtrUpAlpha
11846   \string<\glsxtrUpBeta
11847   \string<\glsxtrUpGamma
11848   \string<\glsxtrUpDelta
11849   \string<\glsxtrUpEpsilon
11850   \string<\glsxtrUpZeta
11851   \string<\glsxtrUpEta
11852   \string<\glsxtrUpTheta
11853   \string<\glsxtrUpIota
11854   \string<\glsxtrUpKappa
11855   \string<\glsxtrUpLambda
11856   \string<\glsxtrUpMu
11857   \string<\glsxtrUpNu
```

```

11858 \string<\glxtrUpXi
11859 \string<\glxtrUpOmicron
11860 \string<\glxtrUpPi
11861 \string<\glxtrUpRho
11862 \string<\glxtrUpSigma
11863 \string<\glxtrUpTau
11864 \string<\glxtrUpUpsilon
11865 \string<\glxtrUpPhi
11866 \string<\glxtrUpChi
11867 \string<\glxtrUpPsi
11868 \string<\glxtrUpOmega
11869 }

```

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.

```

11870 \newcommand*{\glxtrMathItalicGreekIrules}{%
11871 \glxtrMathItalicAlpha
11872 \string<\glxtrMathItalicBeta
11873 \string<\glxtrMathItalicGamma
11874 \string<\glxtrMathItalicDelta
11875 \string<\glxtrMathItalicEpsilon
11876 \string<\glxtrUpDigamma
11877 \string<\glxtrMathItalicZeta
11878 \string<\glxtrMathItalicEta
11879 \string<\glxtrMathItalicTheta
11880 \string<\glxtrMathItalicIota
11881 \string<\glxtrMathItalicKappa
11882 \string<\glxtrMathItalicLambda
11883 \string<\glxtrMathItalicMu
11884 \string<\glxtrMathItalicNu
11885 \string<\glxtrMathItalicXi
11886 \string<\glxtrMathItalicOmicron
11887 \string<\glxtrMathItalicPi
11888 \string<\glxtrMathItalicRho
11889 \string<\glxtrMathItalicSigma
11890 \string<\glxtrMathItalicTau
11891 \string<\glxtrMathItalicUpsilon
11892 \string<\glxtrMathItalicPhi
11893 \string<\glxtrMathItalicChi
11894 \string<\glxtrMathItalicPsi
11895 \string<\glxtrMathItalicOmega
11896 }

```

alicGreekIIrules Doesn't include digamma.

```

11897 \newcommand*{\glxtrMathItalicGreekIIrules}{%
11898 \glxtrMathItalicAlpha
11899 \string<\glxtrMathItalicBeta
11900 \string<\glxtrMathItalicGamma
11901 \string<\glxtrMathItalicDelta

```

```

11902 \string<\glxtrMathItalicEpsilon
11903 \string<\glxtrMathItalicZeta
11904 \string<\glxtrMathItalicEta
11905 \string<\glxtrMathItalicTheta
11906 \string<\glxtrMathItalicIota
11907 \string<\glxtrMathItalicKappa
11908 \string<\glxtrMathItalicLambda
11909 \string<\glxtrMathItalicMu
11910 \string<\glxtrMathItalicNu
11911 \string<\glxtrMathItalicXi
11912 \string<\glxtrMathItalicOmicron
11913 \string<\glxtrMathItalicPi
11914 \string<\glxtrMathItalicRho
11915 \string<\glxtrMathItalicSigma
11916 \string<\glxtrMathItalicTau
11917 \string<\glxtrMathItalicUpsilon
11918 \string<\glxtrMathItalicPhi
11919 \string<\glxtrMathItalicChi
11920 \string<\glxtrMathItalicPsi
11921 \string<\glxtrMathItalicOmega
11922 }

```

UpperGreekIrules Upper case only (includes upright digamma).

```

11923 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
11924 \glshex 1D6E2% upper case alpha (maths italic)
11925 \string<\glshex 1D6E3% upper case beta (maths italic)
11926 \string<\glshex 1D6E4% upper case gamma (maths italic)
11927 \string<\glshex 1D6E5% upper case delta (maths italic)
11928 \string<\glshex 1D6E6% upper case epsilon (maths italic)
11929 \string<\glshex 03DC% upper case digamma
11930 \string<\glshex 1D6E7% upper case zeta (maths italic)
11931 \string<\glshex 1D6E8% upper case eta (maths italic)
11932 \string<\glshex 1D6E9% upper case theta (maths italic)
11933 \string<\glshex 1D6F3% upper case theta variant (maths italic)
11934 \string<\glshex 1D6EA% upper case iota (maths italic)
11935 \string<\glshex 1D6EB% upper case kappa (maths italic)
11936 \string<\glshex 1D6EC% upper case lambda (maths italic)
11937 \string<\glshex 1D6ED% upper case mu (maths italic)
11938 \string<\glshex 1D6EE% upper case nu (maths italic)
11939 \string<\glshex 1D6EF% upper case xi (maths italic)
11940 \string<\glshex 1D6F0% upper case omicron (maths italic)
11941 \string<\glshex 1D6F1% upper case pi (maths italic)
11942 \string<\glshex 1D6F2% upper case rho (maths italic)
11943 \string<\glshex 1D6F4% upper case sigma (maths italic)
11944 \string<\glshex 1D6F5% upper case tau (maths italic)
11945 \string<\glshex 1D6F6% upper case upsilon (maths italic)
11946 \string<\glshex 1D6F7% upper case phi (maths italic)
11947 \string<\glshex 1D6F8% upper case chi (maths italic)
11948 \string<\glshex 1D6F9% upper case psi (maths italic)

```

```

11949 \string<\glshex 1D6FA% upper case omega (maths italic)
11950 }

```

perGreekIIrules Upper case only (doesn't include upright digamma).

```

11951 \newcommand*{\glxtrMathItalicUpperGreekIIrules}{%
11952 \glshex 1D6E2% upper case alpha (maths italic)
11953 \string<\glshex 1D6E3% upper case beta (maths italic)
11954 \string<\glshex 1D6E4% upper case gamma (maths italic)
11955 \string<\glshex 1D6E5% upper case delta (maths italic)
11956 \string<\glshex 1D6E6% upper case epsilon (maths italic)
11957 \string<\glshex 1D6E7% upper case zeta (maths italic)
11958 \string<\glshex 1D6E8% upper case eta (maths italic)
11959 \string<\glshex 1D6E9% upper case theta (maths italic)
11960 \string=<\glshex 1D6F3% upper case theta variant (maths italic)
11961 \string<\glshex 1D6EA% upper case iota (maths italic)
11962 \string<\glshex 1D6EB% upper case kappa (maths italic)
11963 \string<\glshex 1D6EC% upper case lambda (maths italic)
11964 \string<\glshex 1D6ED% upper case mu (maths italic)
11965 \string<\glshex 1D6EE% upper case nu (maths italic)
11966 \string<\glshex 1D6EF% upper case xi (maths italic)
11967 \string<\glshex 1D6F0% upper case omicron (maths italic)
11968 \string<\glshex 1D6F1% upper case pi (maths italic)
11969 \string<\glshex 1D6F2% upper case rho (maths italic)
11970 \string<\glshex 1D6F4% upper case sigma (maths italic)
11971 \string<\glshex 1D6F5% upper case tau (maths italic)
11972 \string<\glshex 1D6F6% upper case upsilon (maths italic)
11973 \string<\glshex 1D6F7% upper case phi (maths italic)
11974 \string<\glshex 1D6F8% upper case chi (maths italic)
11975 \string<\glshex 1D6F9% upper case psi (maths italic)
11976 \string<\glshex 1D6FA% upper case omega (maths italic)
11977 }

```

owerGreekIrules Lower case only (includes upright digamma).

```

11978 \newcommand*{\glxtrMathItalicLowerGreekIrules}{%
11979 \glshex 1D6FC% lower case alpha (maths italic)
11980 \string<\glshex 1D6FD% lower case beta (maths italic)
11981 \string<\glshex 1D6FE% lower case gamma (maths italic)
11982 \string<\glshex 1D6FF% lower case delta (maths italic)
11983 \string<\glshex 1D700% lower case epsilon (maths italic)
11984 \string=<\glshex 1D716% lower case epsilon variant (maths italic)
11985 \string<\glshex 03DD% lower case digamma
11986 \string<\glshex 1D701% lower case zeta (maths italic)
11987 \string<\glshex 1D702% lower case eta (maths italic)
11988 \string<\glshex 1D703% lower case theta (maths italic)
11989 \string=<\glshex 1D717% lower case theta variant (maths italic)
11990 \string<\glshex 1D704% lower case iota (maths italic)
11991 \string<\glshex 1D705% lower case kappa (maths italic)
11992 \string=<\glshex 1D718% lower case kappa variant (maths italic)
11993 \string<\glshex 1D706% lower case lambda (maths italic)

```

```

11994 \string<\glshex 1D707% lower case mu (maths italic)
11995 \string<\glshex 1D708% lower case nu (maths italic)
11996 \string<\glshex 1D709% lower case xi (maths italic)
11997 \string<\glshex 1D70A% lower case omicron (maths italic)
11998 \string<\glshex 1D70B% lower case pi (maths italic)
11999 \string=\glshex 1D71B% lower case pi variant (maths italic)
12000 \string<\glshex 1D70C% lower case rho (maths italic)
12001 \string=\glshex 1D71A% lower case rho variant (maths italic)
12002 \string<\glshex 1D70D% lower case final sigma (maths italic)
12003 \string=\glshex 1D70E% lower case sigma (maths italic)
12004 \string<\glshex 1D70F% lower case tau (maths italic)
12005 \string<\glshex 1D710% lower case upsilon (maths italic)
12006 \string<\glshex 1D711% lower case phi (maths italic)
12007 \string=\glshex 1D719% lower case phi variant (maths italic)
12008 \string<\glshex 1D712% lower case chi (maths italic)
12009 \string<\glshex 1D713% lower case psi (maths italic)
12010 \string<\glshex 1D714% lower case omega (maths italic)
12011 }

```

LowerGreekIIrules Lower case only (doesn't includes upright digamma).

```

12012 \newcommand*{\glxtrMathItalicLowerGreekIIrules}{%
12013 \glshex 1D6FC% lower case alpha (maths italic)
12014 \string<\glshex 1D6FD% lower case beta (maths italic)
12015 \string<\glshex 1D6FE% lower case gamma (maths italic)
12016 \string<\glshex 1D6FF% lower case delta (maths italic)
12017 \string<\glshex 1D700% lower case epsilon (maths italic)
12018 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12019 \string<\glshex 1D701% lower case zeta (maths italic)
12020 \string<\glshex 1D702% lower case eta (maths italic)
12021 \string<\glshex 1D703% lower case theta (maths italic)
12022 \string=\glshex 1D717% lower case theta variant (maths italic)
12023 \string<\glshex 1D704% lower case iota (maths italic)
12024 \string<\glshex 1D705% lower case kappa (maths italic)
12025 \string=\glshex 1D718% lower case kappa variant (maths italic)
12026 \string<\glshex 1D706% lower case lambda (maths italic)
12027 \string<\glshex 1D707% lower case mu (maths italic)
12028 \string<\glshex 1D708% lower case nu (maths italic)
12029 \string<\glshex 1D709% lower case xi (maths italic)
12030 \string<\glshex 1D70A% lower case omicron (maths italic)
12031 \string<\glshex 1D70B% lower case pi (maths italic)
12032 \string=\glshex 1D71B% lower case pi variant (maths italic)
12033 \string<\glshex 1D70C% lower case rho (maths italic)
12034 \string=\glshex 1D71A% lower case rho variant (maths italic)
12035 \string<\glshex 1D70D% lower case final sigma (maths italic)
12036 \string=\glshex 1D70E% lower case sigma (maths italic)
12037 \string<\glshex 1D70F% lower case tau (maths italic)
12038 \string<\glshex 1D710% lower case upsilon (maths italic)
12039 \string<\glshex 1D711% lower case phi (maths italic)
12040 \string=\glshex 1D719% lower case phi variant (maths italic)

```

```

12041 \string<\glshex 1D712% lower case chi (maths italic)
12042 \string<\glshex 1D713% lower case psi (maths italic)
12043 \string<\glshex 1D714% lower case omega (maths italic)
12044 }

```

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

```

12045 \newcommand*{\glxtrMathGreekIrules}{%
12046 \glxtrMathItalicAlpha
12047 \string;\glxtrUpAlpha
12048 \string<\glxtrMathItalicBeta
12049 \string;\glxtrUpBeta
12050 \string<\glxtrMathItalicGamma
12051 \string;\glxtrUpGamma
12052 \string<\glxtrMathItalicDelta
12053 \string;\glxtrUpDelta
12054 \string<\glxtrMathItalicEpsilon
12055 \string;\glxtrUpEpsilon
12056 \string<\glxtrUpDigamma
12057 \string<\glxtrMathItalicZeta
12058 \string;\glxtrUpZeta
12059 \string<\glxtrMathItalicEta
12060 \string;\glxtrUpEta
12061 \string<\glxtrMathItalicTheta
12062 \string;\glxtrUpTheta
12063 \string<\glxtrMathItalicIota
12064 \string;\glxtrUpIota
12065 \string<\glxtrMathItalicKappa
12066 \string;\glxtrUpKappa
12067 \string<\glxtrMathItalicLambda
12068 \string;\glxtrUpLambda
12069 \string<\glxtrMathItalicMu
12070 \string;\glxtrUpMu
12071 \string<\glxtrMathItalicNu
12072 \string;\glxtrUpNu
12073 \string<\glxtrMathItalicXi
12074 \string;\glxtrUpXi
12075 \string<\glxtrMathItalicOmicron
12076 \string;\glxtrUpOmicron
12077 \string<\glxtrMathItalicPi
12078 \string;\glxtrUpPi
12079 \string<\glxtrMathItalicRho
12080 \string;\glxtrUpRho
12081 \string<\glxtrMathItalicSigma
12082 \string;\glxtrUpSigma
12083 \string<\glxtrMathItalicTau
12084 \string;\glxtrUpTau
12085 \string<\glxtrMathItalicUpsilon
12086 \string;\glxtrUpUpsilon
12087 \string<\glxtrMathItalicPhi

```

```

12088 \string;\glxtrUpPhi
12089 \string<\glxtrMathItalicChi
12090 \string;\glxtrUpChi
12091 \string<\glxtrMathItalicPsi
12092 \string;\glxtrUpPsi
12093 \string<\glxtrMathItalicOmega
12094 \string;\glxtrUpOmega
12095 }

```

athGreekIIrules Includes both upright and italic (digamma not included).

```

12096 \newcommand*{\glxtrMathGreekIIrules}{%
12097 \glxtrMathItalicAlpha
12098 \string;\glxtrUpAlpha
12099 \string<\glxtrMathItalicBeta
12100 \string;\glxtrUpBeta
12101 \string<\glxtrMathItalicGamma
12102 \string;\glxtrUpGamma
12103 \string<\glxtrMathItalicDelta
12104 \string;\glxtrUpDelta
12105 \string<\glxtrMathItalicEpsilon
12106 \string;\glxtrUpEpsilon
12107 \string<\glxtrMathItalicZeta
12108 \string;\glxtrUpZeta
12109 \string<\glxtrMathItalicEta
12110 \string;\glxtrUpEta
12111 \string<\glxtrMathItalicTheta
12112 \string;\glxtrUpTheta
12113 \string<\glxtrMathItalicIota
12114 \string;\glxtrUpIota
12115 \string<\glxtrMathItalicKappa
12116 \string;\glxtrUpKappa
12117 \string<\glxtrMathItalicLambda
12118 \string;\glxtrUpLambda
12119 \string<\glxtrMathItalicMu
12120 \string;\glxtrUpMu
12121 \string<\glxtrMathItalicNu
12122 \string;\glxtrUpNu
12123 \string<\glxtrMathItalicXi
12124 \string;\glxtrUpXi
12125 \string<\glxtrMathItalicOmicron
12126 \string;\glxtrUpOmicron
12127 \string<\glxtrMathItalicPi
12128 \string;\glxtrUpPi
12129 \string<\glxtrMathItalicRho
12130 \string;\glxtrUpRho
12131 \string<\glxtrMathItalicSigma
12132 \string;\glxtrUpSigma
12133 \string<\glxtrMathItalicTau
12134 \string;\glxtrUpTau

```



```

12135 \string<\glxtrMathItalicUpsilon
12136 \string;\glxtrUpUpsilon
12137 \string<\glxtrMathItalicPhi
12138 \string;\glxtrUpPhi
12139 \string<\glxtrMathItalicChi
12140 \string;\glxtrUpChi
12141 \string<\glxtrMathItalicPsi
12142 \string;\glxtrUpPsi
12143 \string<\glxtrMathItalicOmega
12144 \string;\glxtrUpOmega
12145 }

```

\glxtrUpAlpha

```

12146 \newcommand*{\glxtrUpAlpha}{%
12147 \glshex 03B1,% lower case alpha
12148 \glshex 0391% upper case alpha
12149 }

```

\glxtrUpBeta

```

12150 \newcommand*{\glxtrUpBeta}{%
12151 \glshex 03B2,% lower case beta
12152 \glshex 0392% upper case beta
12153 }

```

\glxtrUpGamma

```

12154 \newcommand*{\glxtrUpGamma}{%
12155 \glshex 03B3,% lower case gamma
12156 \glshex 0393% upper case gamma
12157 }

```

\glxtrUpDelta

```

12158 \newcommand*{\glxtrUpDelta}{%
12159 \glshex 03B4,% lower case delta
12160 \glshex 0394% upper case delta
12161 }

```

glxtrUpEpsilon

```

12162 \newcommand*{\glxtrUpEpsilon}{%
12163 \glshex 03B5% lower case epsilon
12164 \string=\glshex 03F5,% lower case epsilon variant
12165 \glshex 0395% upper case epsilon
12166 }

```

glxtrUpDigamma

```

12167 \newcommand*{\glxtrUpDigamma}{%
12168 \glshex 03DD,% lower case digamma
12169 \glshex 03DC% upper case digamma
12170 }

```

`\glxtrUpZeta`

```
12171 \newcommand*{\glxtrUpZeta}{%
12172   \glshex 03B6,% lower case zeta
12173   \glshex 0396% upper case zeta
12174 }
```

`\glxtrUpEta`

```
12175 \newcommand*{\glxtrUpEta}{%
12176   \glshex 03B7,% lower case eta
12177   \glshex 0397% upper case eta
12178 }
```

`\glxtrUpTheta`

```
12179 \newcommand*{\glxtrUpTheta}{%
12180   \glshex 03B8% lower case theta
12181   \string=\glshex 03D1,% lower case theta variant
12182   \glshex 0398% upper case theta
12183 }
```

`\glxtrUpIota`

```
12184 \newcommand*{\glxtrUpIota}{%
12185   \glshex 03B9,% lower case iota
12186   \glshex 0399% upper case iota
12187 }
```

`\glxtrUpKappa`

```
12188 \newcommand*{\glxtrUpKappa}{%
12189   \glshex 03BA% lower case kappa
12190   \string=\glshex 03F0,% lower case kappa variant
12191   \glshex 039A% upper case kappa
12192 }
```

`\glxtrUpLambda`

```
12193 \newcommand*{\glxtrUpLambda}{%
12194   \glshex 03BB,% lower lambda
12195   \glshex 039B% upper case lambda
12196 }
```

`\glxtrUpMu`

```
12197 \newcommand*{\glxtrUpMu}{%
12198   \glshex 03BC,% lower case mu
12199   \glshex 039C% upper case mu
12200 }
```

`\glxtrUpNu`

```
12201 \newcommand*{\glxtrUpNu}{%
12202   \glshex 03BD,% lower case nu
12203   \glshex 039D% upper case nu
12204 }
```

\glxtrUpXi

```
12205 \newcommand*{\glxtrUpXi}{%
12206   \glshex 03BE,% lower case xi
12207   \glshex 039E% upper case xi
12208 }
```

glxtrUpOmicron

```
12209 \newcommand*{\glxtrUpOmicron}{%
12210   \glshex 03BF,% lower case omicron
12211   \glshex 039F% upper case omicron
12212 }
```

\glxtrUpPi

```
12213 \newcommand*{\glxtrUpPi}{%
12214   \glshex 03C0% lower case pi
12215   \string=\glshex 03D6,% lower case pi variant
12216   \glshex 03A0% upper case pi
12217 }
```

\glxtrUpRho

```
12218 \newcommand*{\glxtrUpRho}{%
12219   \glshex 03C1% lower case rho
12220   \string=\glshex 03F1,% lower case rho variant
12221   \glshex 03A1% upper case rho
12222 }
```

\glxtrUpSigma

```
12223 \newcommand*{\glxtrUpSigma}{%
12224   \glshex 03C2% lower case sigma
12225   \string=\glshex 03C3,% lower case sigma
12226   \glshex 03A3% upper case sigma
12227 }
```

\glxtrUpTau

```
12228 \newcommand*{\glxtrUpTau}{%
12229   \glshex 03C4,% lower case tau
12230   \glshex 03A4% upper case tau
12231 }
```

glxtrUpUpsilon

```
12232 \newcommand*{\glxtrUpUpsilon}{%
12233   \glshex 03C5,% lower case upsilon
12234   \glshex 03A5% upper case upsilon
12235 }
```

\glxtrUpPhi

```
12236 \newcommand*{\glxtrUpPhi}{%
12237   \glshex 03C6% lower case phi
```

```

12238 \string=\glshex 03D5,% lower case phi variant
12239 \glshex 03A6% upper case phi
12240 }

```

$\backslash\mathrm{glxtrUpChi}$

```

12241 \newcommand*{\glxtrUpChi}{%
12242 \glshex 03C7,% lower case chi
12243 \glshex 03A7% upper case chi
12244 }

```

$\backslash\mathrm{glxtrUpPsi}$

```

12245 \newcommand*{\glxtrUpPsi}{%
12246 \glshex 03C8,% lower case psi
12247 \glshex 03A8% upper case psi
12248 }

```

$\backslash\mathrm{glxtrUpOmega}$

```

12249 \newcommand*{\glxtrUpOmega}{%
12250 \glshex 03C9,% lower case omega
12251 \glshex 03A9% upper case omega
12252 }

```

$\mathrm{MathItalicAlpha}$

```

12253 \newcommand*{\glxtrMathItalicAlpha}{%
12254 \glshex 1D6FC,% lower case alpha (maths italic)
12255 \glshex 1D6E2% upper case alpha (maths italic)
12256 }

```

$\mathrm{rMathItalicBeta}$

```

12257 \newcommand*{\glxtrMathItalicBeta}{%
12258 \glshex 1D6FD,% lower case beta (maths italic)
12259 \glshex 1D6E3% upper case beta (maths italic)
12260 }

```

$\mathrm{MathItalicGamma}$

```

12261 \newcommand*{\glxtrMathItalicGamma}{%
12262 \glshex 1D6FE,% lower case gamma (maths italic)
12263 \glshex 1D6E4% upper case gamma (maths italic)
12264 }

```

$\mathrm{MathItalicDelta}$

```

12265 \newcommand*{\glxtrMathItalicDelta}{%
12266 \glshex 1D6FF,% lower case delta (maths italic)
12267 \glshex 1D6E5% upper case delta (maths italic)
12268 }

```

$\mathrm{thItalicEpsilon}$

```

12269 \newcommand*{\glxtrMathItalicEpsilon}{%

```

```

12270 \glshex 1D700% lower case epsilon (maths italic)
12271 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
12272 \glshex 1D6E6% upper case epsilon (maths italic)
12273 }

```

rMathItalicZeta

```

12274 \newcommand*{\glxtrMathItalicZeta}{%
12275 \glshex 1D701,% lower case zeta (maths italic)
12276 \glshex 1D6E7% upper case zeta (maths italic)
12277 }

```

trMathItalicEta

```

12278 \newcommand*{\glxtrMathItalicEta}{%
12279 \glshex 1D702,% lower case eta (maths italic)
12280 \glshex 1D6E8% upper case eta (maths italic)
12281 }

```

MathItalicTheta

```

12282 \newcommand*{\glxtrMathItalicTheta}{%
12283 \glshex 1D703% lower case theta (maths italic)
12284 \string=\glshex 1D717,% lower case theta variant (maths italic)
12285 \glshex 1D6E9% upper case theta (maths italic)
12286 \string=\glshex 1D6F3% upper case theta variant (maths italic)
12287 }

```

rMathItalicIota

```

12288 \newcommand*{\glxtrMathItalicIota}{%
12289 \glshex 1D704,% lower case iota (maths italic)
12290 \glshex 1D6EA% upper case iota (maths italic)
12291 }

```

MathItalicKappa

```

12292 \newcommand*{\glxtrMathItalicKappa}{%
12293 \glshex 1D705% lower case kappa (maths italic)
12294 \string=\glshex 1D718,% lower case kappa variant (maths italic)
12295 \glshex 1D6EB% upper case kappa (maths italic)
12296 }

```

athItalicLambda

```

12297 \newcommand*{\glxtrMathItalicLambda}{%
12298 \glshex 1D706,% lower case lambda (maths italic)
12299 \glshex 1D6EC% upper case lambda (maths italic)
12300 }

```

xtrMathItalicMu

```

12301 \newcommand*{\glxtrMathItalicMu}{%
12302 \glshex 1D707,% lower case mu (maths italic)
12303 \glshex 1D6ED% upper case mu (maths italic)
12304 }

```

xtrMathItalicNu

```
12305 \newcommand*{\glxtrMathItalicNu}{%
12306   \glshex 1D708,% lower case nu (maths italic)
12307   \glshex 1D6EE% upper case nu (maths italic)
12308 }
```

xtrMathItalicXi

```
12309 \newcommand*{\glxtrMathItalicXi}{%
12310   \glshex 1D709,% lower case xi (maths italic)
12311   \glshex 1D6EF% upper case xi (maths italic)
12312 }
```

thItalicOmicron

```
12313 \newcommand*{\glxtrMathItalicOmicron}{%
12314   \glshex 1D70A,% lower case omicron (maths italic)
12315   \glshex 1D6F0% upper case omicron (maths italic)
12316 }
```

xtrMathItalicPi

```
12317 \newcommand*{\glxtrMathItalicPi}{%
12318   \glshex 1D70B% lower case pi (maths italic)
12319   \string=\glshex 1D71B,% lower case pi variant (maths italic)
12320   \glshex 1D6F1% upper case pi (maths italic)
12321 }
```

trMathItalicRho

```
12322 \newcommand*{\glxtrMathItalicRho}{%
12323   \glshex 1D70C% lower case rho (maths italic)
12324   \string=\glshex 1D71A,% lower case rho variant (maths italic)
12325   \glshex 1D6F2% upper case rho (maths italic)
12326 }
```

MathItalicSigma

```
12327 \newcommand*{\glxtrMathItalicSigma}{%
12328   \glshex 1D70D% lower case final sigma (maths italic)
12329   \string=\glshex 1D70E,% lower case sigma (maths italic)
12330   \glshex 1D6F4% upper case sigma (maths italic)
12331 }
```

trMathItalicTau

```
12332 \newcommand*{\glxtrMathItalicTau}{%
12333   \glshex 1D70F,% lower case tau (maths italic)
12334   \glshex 1D6F5% upper case tau (maths italic)
12335 }
```

thItalicUpsilon

```
12336 \newcommand*{\glxtrMathItalicUpsilon}{%
12337   \glshex 1D710,% lower case upsilon (maths italic)
```

```

12338 \glshex 1D6F6% upper case upsilon (maths italic)
12339 }

```

trMathItalicPhi

```

12340 \newcommand*{\glsxtrMathItalicPhi}{%
12341 \glshex 1D711% lower case phi (maths italic)
12342 \string=\glshex 1D719,% lower case phi variant (maths italic)
12343 \glshex 1D6F7% upper case phi (maths italic)
12344 }

```

trMathItalicChi

```

12345 \newcommand*{\glsxtrMathItalicChi}{%
12346 \glshex 1D712,% lower case chi (maths italic)
12347 \glshex 1D6F8% upper case chi (maths italic)
12348 }

```

trMathItalicPsi

```

12349 \newcommand*{\glsxtrMathItalicPsi}{%
12350 \glshex 1D713,% lower case psi (maths italic)
12351 \glshex 1D6F9% upper case psi (maths italic)
12352 }

```

MathItalicOmega

```

12353 \newcommand*{\glsxtrMathItalicOmega}{%
12354 \glshex 1D714,% lower case omega (maths italic)
12355 \glshex 1D6FA% upper case omega (maths italic)
12356 }

```

thItalicPartial

```

12357 \newcommand*{\glsxtrMathItalicPartial}{%
12358 \glshex 1D715% partial differential (maths italic)
12359 }

```

MathItalicNabla

```

12360 \newcommand*{\glsxtrMathItalicNabla}{%
12361 \glshex 1D6FB% nabla (maths italic)
12362 }

```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```

12363 \newcommand*{\glsxtrdigitrules}{%
12364 0\string=\glshex 2080\string=\glshex 2070
12365 \string<1\string=\glshex 2081\string=\glshex 00B9
12366 \string<2\string=\glshex 2082\string=\glshex 00B2
12367 \string<3\string=\glshex 2083\string=\glshex 00B3
12368 \string<4\string=\glshex 2084\string=\glshex 2074
12369 \string<5\string=\glshex 2085\string=\glshex 2075
12370 \string<6\string=\glshex 2086\string=\glshex 2076
12371 \string<7\string=\glshex 2087\string=\glshex 2077

```

```

12372 \string<8\string=\glshex 2088\string=\glshex 2078
12373 \string<9\string=\glshex 2089\string=\glshex 2079
12374 }

```

BasicDigitrules Digits from the Basic Latin set.

```

12375 \newcommand*{\glxtrBasicDigitrules}{%
12376 0\string<1\string<2\string<3\string<4%
12377 \string<5\string<6\string<7\string<8\string<9%
12378 }

```

criptDigitrules Subscript digits.

```

12379 \newcommand*{\glxtrSubScriptDigitrules}{%
12380 \glshex 2080% subscript 0
12381 \string<\glshex 2081% subscript 1
12382 \string<\glshex 2082% subscript 2
12383 \string<\glshex 2083% subscript 3
12384 \string<\glshex 2084% subscript 4
12385 \string<\glshex 2085% subscript 5
12386 \string<\glshex 2086% subscript 6
12387 \string<\glshex 2087% subscript 7
12388 \string<\glshex 2088% subscript 8
12389 \string<\glshex 2089% subscript 9
12390 }

```

criptDigitrules Superscript digits.

```

12391 \newcommand*{\glxtrSuperScriptDigitrules}{%
12392 \glshex 2070% superscript 0
12393 \string<\glshex 00B9% superscript 1
12394 \string<\glshex 00B2% superscript 2
12395 \string<\glshex 00B3% superscript 3
12396 \string<\glshex 2074% superscript 4
12397 \string<\glshex 2075% superscript 5
12398 \string<\glshex 2076% superscript 6
12399 \string<\glshex 2077% superscript 7
12400 \string<\glshex 2078% superscript 8
12401 \string<\glshex 2079% superscript 9
12402 }

```

trfractionrules Vulgar fractions.

```

12403 \newcommand*{\glxtrfractionrules}{%
12404 \glshex 215F% fraction numerator one (1/)
12405 \string<\glshex 2189% zero thirds (0/3 = 0)
12406 \string<\glshex 2152% one tenth (1/10 = 0.1)
12407 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
12408 \string<\glshex 215B% one eighth (1/8 = 0.125)
12409 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
12410 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
12411 \string<\glshex 2155% one fifth (1/5 = 0.2)
12412 \string<\glshex 00BC% one quarter (1/4 = 0.25)

```



```

12413 \string<\glshex 2153% one third ( $1/3 \sim 0.333$ )
12414 \string<\glshex 215C% three eighths ( $3/8 = 0.375$ )
12415 \string<\glshex 2156% two fifths ( $2/5 = 0.4$ )
12416 \string<\glshex 00BD% one half ( $1/2 = 0.5$ )
12417 \string<\glshex 2157% three fifths ( $3/5 = 0.6$ )
12418 \string<\glshex 215D% five eighths ( $5/8 = 0.625$ )
12419 \string<\glshex 2154% two thirds ( $2/3 \sim 0.667$ )
12420 \string<\glshex 00BE% three quarters ( $3/4 = 0.75$ )
12421 \string<\glshex 2158% four fifths ( $4/5 = 0.8$ )
12422 \string<\glshex 215A% five sixths ( $5/6 \sim 0.833$ )
12423 \string<\glshex 215E% seven eighths ( $7/8 = 0.875$ )
12424 }

```

Check for scripts associated with the document dialects.

```

12425 \renewcommand{\@glxtrdialecthook}{%
12426   \ifundef\CurrentTrackedScript
12427   {%
12428     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
12429     {%
12430       \edef\CurrentTrackedScript{%
12431         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
12432       }%
12433     }%
12434   }%
12435   {}%
12436   \ifdef\CurrentTrackedScript
12437   {%
12438     \let\CurrentTrackedTag\CurrentTrackedScript
12439     \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
12440     {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
12441     {}%
12442   }%
12443   {}%
12444 }

```

If `\glxtr@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.

```

12445 \ifdef\glxtr@loaddialect
12446 {%
12447   \@ifpackageloaded{tracklang}
12448   {%
12449     \AnyTrackedLanguages
12450     {%
12451       \ForEachTrackedDialect{\this@dialect}{\glxtr@loaddialect}%
12452     }%
12453   }%
12454 }
12455 {}

```

12456 }
12457 {}

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:

```
12458 \NeedsTeXFormat{LaTeX2e}
```

```
12459 \ProvidesPackage{glossaries-extra-stylemods}[2018/02/26 v1.27 (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
```

```
12460 \newcommand*{\@glxtr@loadstyles}{}
```

all Provide all known styles.

```
12461 \DeclareOption{all}{%
```

```
12462   \appto\@glxtr@loadstyles{%
```

```
12463     \RequirePackage{glossary-inline}%
```

```
12464     \RequirePackage{glossary-list}%
```

```
12465     \RequirePackage{glossary-tree}%
```

```
12466     \RequirePackage{glossary-mcols}%
```

```
12467     \RequirePackage{glossary-long}%
```

```
12468     \RequirePackage{glossary-longragged}%
```

```
12469     \RequirePackage{glossary-longbooktabs}%
```

```
12470     \RequirePackage{glossary-super}%
```

```
12471     \RequirePackage{glossary-superragged}%
```

```
12472     \RequirePackage{glossary-bookindex}%
```

```
12473   }
```

```
12474 }
```

```
12475 \DeclareOption*{%
```

```
12476   \IfFileExists{glossary-\CurrentOption.sty}
```

```
12477   {\eappto\@glxtr@loadstyles{%
```

```
12478     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
```

```
12479   }%
```

```
12480   {%
```

```
12481     \PackageError{glossaries-extra-styles}%
```

```

12482     {Unknown option ‘\CurrentOption’}{}%
12483   }%
12484 }

```

Process the package options:

```
12485 \ProcessOptions
```

Load the required packages:

```
12486 \@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`.

```
12487 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

`ewglossarystyle`

```

12488 \providecommand{\renewglossarystyle}[2]{%
12489   \ifcsundef{@glsstyle@#1}%
12490   {%
12491     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
12492   }%
12493   {%
12494     \csdef{@glsstyle@#1}{#2}%
12495   }%
12496 }

```

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.

```

12497 \ifdef{@glsstyle@listdotted}
12498 {%
12499   \renewglossarystyle{listdotted}{%
12500     \setglossarystyle{list}%
12501     \renewcommand*{\glossentry}[2]{%
12502       \item[]\makebox[\glslistdottedwidth][l]{%
12503         \glsentryitem{##1}%
12504         \glstarget{##1}{\glossentryname{##1}}%
12505         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
12506         \glossentrydesc{##1}\glspostdescription}%
12507     \renewcommand*{\subglossentry}[3]{%
12508       \item[]\makebox[\glslistdottedwidth][l]{%
12509         \glssubentryitem{##2}%
12510         \glstarget{##2}{\glossentryname{##2}}%
12511         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
12512         \glossentrydesc{##2}\glspostdescription}%
12513   }

```

```
12514 }
12515 {%
```

Assume the style isn't required if it hasn't already been defined.

```
12516 }
```

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.

```
12517 \ifdef{\@glsstyle@list}
12518 {%
```

listprelocation Space before number list for top-level entries.

```
12519 \newcommand{\glslistprelocation}{\glsxtrprelocation}
```

childprelocation Space before number list for child entries.

```
12520 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
12521 \newcommand{\glslistchildpostlocation}{.}
```

Redefine list to use these commands.

```
12522 \renewglossarystyle{list}{%
12523 \renewenvironment{theglossary}%
12524 {\begin{description}}{\end{description}}%
12525 \renewcommand*{\glossaryheader}{}%
12526 \renewcommand*{\glsgroupheading}[1]{}%
12527 \renewcommand*{\glossentry}[2]{%
12528 \item[\glsentryitem{##1}]%
12529 \glstarget{##1}{\glossentryname{##1}}]
12530 \glossentrydesc{##1}\glspostdescription\glslistprelocation ##2}%
12531 \renewcommand*{\subglossentry}[3]{%
12532 \glssubentryitem{##2}%
12533 \glstarget{##2}{\strut}\space
12534 \glossentrydesc{##2}\glspostdescription
12535 \glslistchildprelocation ##3\glslistchildpostlocation}%
12536 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
12537 }
12538 }
12539 }
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
12540 \ifdef{\@glsstyle@altlist}
12541 {%
12542 \renewglossarystyle{altlist}{%
12543 \setglossarystyle{list}%
12544 \renewcommand*{\glossentry}[2]{%
12545 \item[\glsentryitem{##1}]%
```

```

12546      \glstarget{##1}{\glossentryname{##1}}}%
12547      \mbox{}\par\nobreak\@afterheading
12548      \glossentrydesc{##1}\glspostdescription\glslistprelocation ##2}%
12549      \renewcommand{\subglossentry}[3]{%
12550        \par
12551        \glssubentryitem{##2}%
12552        \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
12553        \glslistchildprelocation ##3}%
12554    }
12555 }
12556 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

12557 \ifdef{\@glsstyle@listgroup}
12558 {%
12559   \renewglossarystyle{listgroup}{%
12560     \setglossarystyle{list}%
12561     \renewcommand*{\glsgroupheading}[1]{%
12562       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}}%
12563       \mbox{}\par\nobreak\@afterheading
12564     }%
12565   }
12566 }
12567 {}

```

Similarly for listhypergroup.

```

12568 \ifdef{\@glsstyle@listhypergroup}
12569 {%
12570   \renewglossarystyle{listhypergroup}{%
12571     \setglossarystyle{list}%
12572     \renewcommand*{\glossaryheader}{%
12573       \glslistnavigationitem{\glsnavigation}}%
12574     \renewcommand*{\glsgroupheading}[1]{%
12575       \item[\glslistgroupheaderfmt
12576         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
12577       \mbox{}\par\nobreak\@afterheading
12578     }%
12579   }
12580 }
12581 {}

```

Similarly for altlistgroup.

```

12582 \ifdef{\@glsstyle@altlistgroup}
12583 {%
12584   \renewglossarystyle{altlistgroup}{%
12585     \setglossarystyle{altlist}%
12586     \renewcommand*{\glsgroupheading}[1]{%
12587       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}}%
12588       \mbox{}\par\nobreak\@afterheading
12589     }%
12590   }

```

```

12591 }
12592 {}

    Similarly for altlisthypergroup.
12593 \ifdef{\@glsstyle@altlisthypergroup}
12594 {%
12595   \renewglossarystyle{altlisthypergroup}{%
12596     \setglossarystyle{altlist}%
12597     \renewcommand*\{glossaryheader\}{%
12598       \glslistnavigationitem{\glsnavigation}}%
12599     \renewcommand*\{glsgroupheading\}[1]{%
12600       \item[\glslistgroupheaderfmt
12601         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
12602       \mbox{}\par\nobreak\@afterheading
12603     }%
12604   }
12605 }
12606 {}

```

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 `\glstrprelocation`.

```

12607 \ifcsdef{@glsstyle@long}
12608 {%
12609   \renewglossarystyle{long}{%
12610     \renewenvironment{theglossary}%
12611       {\begin{longtable}{lp{\glsdescwidth}}}%
12612       {\end{longtable}}%
12613     \renewcommand*\{glossaryheader\}{}%
12614     \renewcommand*\{glsgroupheading\}[1]{}%
12615     \renewcommand{\glossentry}[2]{%
12616       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12617       \glossentrydesc{##1}\glspostdescription
12618       \glstrprelocation ##2\tabularnewline
12619     }%
12620     \renewcommand{\subglossentry}[3]{%
12621       &
12622       \glssubentryitem{##2}%
12623       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
12624       \glstrprelocation ##3\tabularnewline
12625     }%
12626     \ifglsnogroupskip
12627       \renewcommand*\{glsgroupskip\}{}%
12628     \else
12629       \renewcommand*\{glsgroupskip\}{ & \tabularnewline}%
12630     \fi
12631   }

```

```
12632 }
12633 {}
```

Three column style:

```
12634 \ifcsdef{@glsstyle@long3col}
12635 {%
12636   \renewglossarystyle{long3col}{%
12637     \renewenvironment{theglossary}%
12638       {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
12639       {\end{longtable}}}%
12640   \renewcommand*{\glossaryheader}{}%
12641   \renewcommand*{\glsgroupheading}[1]{}%
12642   \renewcommand{\glossentry}[2]{%
12643     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12644     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
12645   }%
12646   \renewcommand{\subglossentry}[3]{%
12647     &
12648     \glssubentryitem{##2}%
12649     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12650     ##3\tabularnewline
12651   }%
```

Conditional needs to be outside of \glsgroupskip otherwise it can cause “Incomplete \iftrue” errors.

```
12652   \ifglsnogroupskip
12653     \renewcommand*{\glsgroupskip}{}%
12654   \else
12655     \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
12656   \fi
12657 }
12658 }
12659 {}
```

Four column style:

```
12660 \ifcsdef{@glsstyle@long4col}
12661 {%
12662   \renewglossarystyle{long4col}{%
12663     \renewenvironment{theglossary}%
12664       {\begin{longtable}{llll}}}%
12665       {\end{longtable}}}%
12666   \renewcommand*{\glossaryheader}{}%
12667   \renewcommand*{\glsgroupheading}[1]{}%
12668   \renewcommand{\glossentry}[2]{%
12669     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12670     \glossentrydesc{##1}\glspostdescription &
12671     \glossentrysymbol{##1} &
12672     ##2\tabularnewline
12673   }%
12674   \renewcommand{\subglossentry}[3]{%
12675     &
```



```

12676     \glssubentryitem{##2}%
12677     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12678     \glossentrysymbol{##2} & ##3\tabularnewline
12679 }%

12680 \ifglsnogroupskip
12681   \renewcommand*{\glsgroupskip}{}%
12682 \else
12683   \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
12684 \fi
12685 }
12686 }
12687 {}

```

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`.

```

12688 \ifcsdef{@glstyle@longragged}
12689 {%
12690   \renewglossarystyle{longragged}{%
12691     \renewenvironment{theglossary}%
12692       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
12693       {\end{longtable}}%
12694     \renewcommand*{\glossaryheader}{}%
12695     \renewcommand*{\glsgroupheading}[1]{}%
12696     \renewcommand{\glossentry}[2]{%
12697       \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12698       \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
12699       \tabularnewline
12700     }%
12701     \renewcommand{\subglossentry}[3]{%
12702       &
12703       \glssubentryitem{##2}%
12704       \glstarget{##2}{\strut}\glossentrydesc{##2}%
12705       \glspostdescription\glxtrprelocation ##3%
12706       \tabularnewline
12707     }%
12708     \ifglsnogroupskip
12709       \renewcommand*{\glsgroupskip}{}%
12710     \else
12711       \renewcommand*{\glsgroupskip}{& \tabularnewline}%
12712     \fi
12713   }
12714 }

```

12715 {}

Three and four column styles don't use \glxstrprelocation since the number list is in its own column.

```
12716 \ifcsdef{@glsstyle@longragged3col}
12717 {%
12718   \renewglossarystyle{longragged3col}{%
12719     \renewenvironment{theglossary}%
12720       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
12721         >{\raggedright}p{\glspagelistwidth}}}%
12722       {\end{longtable}}}%
12723     \renewcommand*{\glossaryheader}{}%
12724     \renewcommand*{\glsgroupheading}[1]{}%
12725     \renewcommand{\glossentry}[2]{%
12726       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12727       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
12728     }%
12729     \renewcommand{\subglossentry}[3]{%
12730       &
12731       \glssubentryitem{##2}%
12732       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12733       ##3\tabularnewline
12734     }%
12735     \ifglsgroupskip
12736       \renewcommand*{\glsgroupskip}{}%
12737     \else
12738       \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
12739     \fi
12740   }
12741 }
12742 {}
```

Four column style:

```
12743 \ifcsdef{@glsstyle@altlongragged4col}
12744 {%
12745   \renewglossarystyle{altlongragged4col}{%
12746     \renewenvironment{theglossary}%
12747       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
12748         >{\raggedright}p{\glspagelistwidth}}}%
12749       {\end{longtable}}}%
12750     \renewcommand*{\glossaryheader}{}%
12751     \renewcommand*{\glsgroupheading}[1]{}%
12752     \renewcommand{\glossentry}[2]{%
12753       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12754       \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
12755       ##2\tabularnewline
12756     }%
12757     \renewcommand{\subglossentry}[3]{%
12758       &
```

```

12759     \glssubentryitem{##2}%
12760     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12761     \glossentrysymbol{##2} & ##3\tabularnewline
12762 }%

12763 \ifglsnogroupskip
12764     \renewcommand*{\glsgroupskip}{}%
12765 \else
12766     \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
12767 \fi
12768 }
12769 }
12770 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment to add the post-description hook. The two column styles need the hard-coded `\space` changed to `\glxtrprelocation`.

```

12771 \ifcsdef{@glstyle@super}
12772 {%
12773     \renewglossarystyle{super}{%
12774         \renewenvironment{theglossary}%
12775             {\tablehead{}\tabletail}%
12776             \begin{supertabular}{lp{\glsdescwidth}}%
12777             {\end{supertabular}}%
12778         \renewcommand*{\glossaryheader}{}%
12779         \renewcommand*{\glsgroupheading}[1]{}%
12780         \renewcommand{\glossentry}[2]{%
12781             \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12782             \glossentrydesc{##1}\glspostdescription
12783             \glxtrprelocation ##2\tabularnewline
12784         }%
12785         \renewcommand{\subglossentry}[3]{%
12786             &
12787             \glssubentryitem{##2}%
12788             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
12789             \glxtrprelocation ##3\tabularnewline
12790         }%
12791         \ifglsnogroupskip
12792             \renewcommand*{\glsgroupskip}{}%
12793         \else
12794             \renewcommand*{\glsgroupskip}{& \tabularnewline}%
12795         \fi
12796     }
12797 }
12798 {}

```

Three column style:

```

12799 \ifcsdef{@glstyle@super3col}

```

```

12800 {%
12801   \renewglossarystyle{super3col}{%
12802     \renewenvironment{theglossary}%
12803       {\tablehead{}\tabletail{}}%
12804       \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
12805       {\end{supertabular}}%
12806     \renewcommand*{\glossaryheader}{}%
12807     \renewcommand*{\glsgroupheading}[1]{}%
12808     \renewcommand{\glossentry}[2]{%
12809       \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12810       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
12811     }%
12812     \renewcommand{\subglossentry}[3]{%
12813       &
12814       \glssubentryitem{##2}%
12815       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12816       ##3\tabularnewline
12817     }%

12818     \ifglsgnogroupskip
12819       \renewcommand*{\glsgroupskip}{}%
12820     \else
12821       \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
12822     \fi
12823   }
12824 }
12825 {}

```

Four column styles:

```

12826 \ifcsdef{@glstyle@super4col}
12827 {%
12828   \renewglossarystyle{super4col}{%
12829     \renewenvironment{theglossary}%
12830       {\tablehead{}\tabletail{}}%
12831       \begin{supertabular}{llll}}}%
12832       \end{supertabular}}%
12833     \renewcommand*{\glossaryheader}{}%
12834     \renewcommand*{\glsgroupheading}[1]{}%
12835     \renewcommand{\glossentry}[2]{%
12836       \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12837       \glossentrydesc{##1}\glspostdescription &
12838       \glossentrysymbol{##1} & ##2\tabularnewline
12839     }%
12840     \renewcommand{\subglossentry}[3]{%
12841       &
12842       \glssubentryitem{##2}%
12843       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12844       \glossentrysymbol{##2} & ##3\tabularnewline
12845     }%

```

```

12846 \ifglsgroupskip
12847 \renewcommand*{\glsgroupskip}{}%
12848 \else
12849 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
12850 \fi
12851 }
12852 }
12853 {}

```

2.6 Super Ragged Styles

The three and four column styles require adjustment for the post-description hook, but not the two column styles. However, the two-column styles need to have `\space` replaced with `\glstrprelocation`.

```

12854 \ifcsdef{@glstyle@superragged}
12855 {%
12856 \renewglossarystyle{superragged}{%
12857 \renewenvironment{theglossary}%
12858 {\tablehead{}}\tabletail{}}%
12859 \begin{supertabular}{1>{\raggedright}p{\glsgdescwidth}}%
12860 {\end{supertabular}}%
12861 \renewcommand*{\glossaryheader}{}%
12862 \renewcommand*{\glsgroupheading}[1]{}%
12863 \renewcommand{\glossentry}[2]{%
12864 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12865 \glossentrydesc{##1}\glspostdescription\glstrprelocation ##2%
12866 \tabularnewline
12867 }%
12868 \renewcommand{\subglossentry}[3]{%
12869 &
12870 \glssubentryitem{##2}%
12871 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
12872 \glstrprelocation ##3%
12873 \tabularnewline
12874 }%
12875 \ifglsgroupskip
12876 \renewcommand*{\glsgroupskip}{}%
12877 \else
12878 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
12879 \fi
12880 }
12881 }
12882 {}

```

Three column style:

```

12883 \ifcsdef{@glstyle@superragged3col}
12884 {%
12885 \renewglossarystyle{superragged3col}{%

```

```

12886 \renewenvironment{theglossary}%
12887   {\tablehead{}\tabletail{}}%
12888   \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
12889     >{\raggedright}p{\glspagelistwidth}}}%
12890   {\end{supertabular}}%
12891 \renewcommand*{\glossaryheader}{}%
12892 \renewcommand*{\glsgroupheading}[1]{}%
12893 \renewcommand{\glossentry}[2]{%
12894   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12895   \glossentrydesc{##1}\glspostdescription &
12896   ##2\tabularnewline
12897 }%
12898 \renewcommand{\subglossentry}[3]{%
12899   &
12900   \glssubentryitem{##2}%
12901   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12902   ##3\tabularnewline
12903 }%
12904 \ifglsgroupskip
12905   \renewcommand*{\glsgroupskip}{}%
12906 \else
12907   \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
12908 \fi
12909 }
12910 }
12911 {}

```

Four columns:

```

12912 \ifcsdef{@glsstyle@altsuperragged4col}
12913 {%
12914   \renewglossarystyle{altsuperragged4col}{%
12915     \renewenvironment{theglossary}%
12916       {\tablehead{}\tabletail{}}%
12917       \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
12918         >{\raggedright}p{\glspagelistwidth}}}%
12919       {\end{supertabular}}%
12920     \renewcommand*{\glossaryheader}{}%
12921     \renewcommand{\glossentry}[2]{%
12922       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12923       \glossentrydesc{##1}\glspostdescription &
12924       \glossentrysymbol{##1} & ##2\tabularnewline
12925     }%
12926     \renewcommand{\subglossentry}[3]{%
12927       &
12928       \glssubentryitem{##2}%
12929       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12930       \glossentrysymbol{##2} & ##3\tabularnewline
12931     }%

```

```

12932 \ifglsnogroupskip
12933 \renewcommand*{\glsgroupskip}{}%
12934 \else
12935 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
12936 \fi
12937 }
12938 }
12939 {}

```

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`.

```

12940 \ifdef{\@glsstyle@inline}
12941 {%
12942 \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
12943 \renewcommand*{\glsinlinedescformat}[3]{%
12944 \space#1\glxtrpostdescription}
12945 \renewcommand*{\glsinlinesubdescformat}[3]{%
12946 #1\glxtrpostdescription}

```

Just use `\glxtrpostdescription` instead of `\glspostdescription`.

```

12947 }
12948 {}

```

2.8 Tree Styles

The index style is redefined so that the space before the number list isn't hard coded.

```

12949 \ifdef{\@glsstyle@index}
12950 {

```

`treeprelocation` The space before the number list for top-level entries. This is shared by the other tree styles.

```

12951 \newcommand*{\glstreeprelocation}{\glxtrprelocation}

```

`childprelocation` The space before the number list for child entries. This is shared by the other tree styles.

```

12952 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}

```

```

12953 \renewglossarystyle{index}{%
12954 \renewenvironment{theglossary}%
12955 {\setlength{\parindent}{0pt}%
12956 \setlength{\parskip}{0pt plus 0.3pt}%
12957 \let\item\glstreeitem
12958 \let\subitem\glstreesubitem

```

```

12959     \let\subsubitem\glstreesubsubitem
12960   }%
12961   {\par}%
12962   \renewcommand*{\glossaryheader}{}%
12963   \renewcommand*{\glsgroupheading}[1]{}%
12964   \renewcommand*{\glossentry}[2]{%
12965     \item\glstentryitem{##1}%
12966     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
12967     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
12968     \glstreepredesc \glossentrydesc{##1}\glspostdescription
12969     \glstreeprelocation ##2%
12970   }%
12971   \renewcommand{\subglossentry}[3]{%
12972     \ifcase##1\relax
12973       \item
12974     \or
12975       \subitem
12976       \glssubentryitem{##2}%
12977     \else
12978       \subsubitem
12979     \fi
12980     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
12981     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
12982     \glstreechildpredesc\glossentrydesc{##2}\glspostdescription
12983     \glstreechildprelocation ##3%
12984   }%
12985   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
12986 }
12987 }
12988 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

12989 \ifdef{\@glsstyle@indexgroup}
12990 {%
12991   \renewglossarystyle{indexgroup}{%
12992     \setglossarystyle{index}%
12993     \renewcommand*{\glsgroupheading}[1]{%
12994       \item\glstreegroupheaderfmt{\glsggetgrouptitle{##1}}%
12995       \nopagebreak\indexspace
12996       \nobreak\@afterheading
12997     }%
12998   }
12999 }
13000 {}

```

Similarly for indexhypergroup.

```

13001 \ifdef{\@glsstyle@indexhypergroup}
13002 {%
13003   \renewglossarystyle{indexhypergroup}{%
13004     \setglossarystyle{index}%

```



```

13005 \renewcommand*{\glossaryheader}{%
13006 \item\glstreenavigationfmt{\glsnavigation}%
13007 \nobreak\@afterheading\indexspace}%
13008 \renewcommand*{\glsgroupheading}[1]{%
13009 \item\glstreegroupheaderfmt
13010 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13011 \nopagebreak\indexspace
13012 \nobreak\@afterheading}%
13013 }%
13014 }
13015 {}

```

Adjust tree style to remove hard coded space before number list.

```

13016 \ifdef{\@glsstyle@tree}
13017 {%
13018 \renewglossarystyle{tree}{%
13019 \renewenvironment{theglossary}%
13020 {\setlength{\parindent}{0pt}%
13021 \setlength{\parskip}{0pt plus 0.3pt}}%
13022 {}%
13023 \renewcommand*{\glossaryheader}{}%
13024 \renewcommand*{\glsgroupheading}[1]{}%
13025 \renewcommand{\glossentry}[2]{%
13026 \hangindent0pt\relax
13027 \parindent0pt\relax
13028 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13029 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
13030 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13031 \glstreeprelocation##2\par
13032 }%
13033 \renewcommand{\subglossentry}[3]{%
13034 \hangindent##1\glstreeindent\relax
13035 \parindent##1\glstreeindent\relax
13036 \ifnum##1=1\relax
13037 \glssubentryitem{##2}%
13038 \fi
13039 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13040 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
13041 \glstreechildpredesc\glossentrydesc{##2}\glspostdescription
13042 \glstreechildprelocation ##3\par
13043 }%
13044 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13045 }%
13046 }
13047 {}

```

The treegroup style is redefined to discourage a page break after the heading.

```

13048 \ifdef{\@glsstyle@treegroup}
13049 {%
13050 \renewglossarystyle{treegroup}{%

```

```

13051 \setglossarystyle{tree}%
13052 \renewcommand{\glsgroupheading}[1]{\par
13053 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
13054 \nopagebreak\indexspace\nobreak\@afterheading}%
13055 }
13056 }
13057 {}

```

Similarly for treehypergroup

```

13058 \ifdef{\@glsstyle@treehypergroup}
13059 {%
13060 \renewglossarystyle{treehypergroup}{%
13061 \setglossarystyle{tree}%
13062 \renewcommand*\{\glossaryheader}{%
13063 \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13064 \nobreak\@afterheading\indexspace}%
13065 \renewcommand*\{\glsgroupheading}[1]{%
13066 \par\noindent
13067 \glstreegroupheaderfmt
13068 {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
13069 \nopagebreak\indexspace\nobreak\@afterheading}%
13070 }
13071 }
13072 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

13073 \ifdef{\@glsstyle@treenoname}
13074 {%
13075 \renewglossarystyle{treenoname}{%
13076 \renewenvironment{theglossary}%
13077 {\setlength{\parindent}{0pt}%
13078 \setlength{\parskip}{0pt plus 0.3pt}}%
13079 {}%
13080 \renewcommand*\{\glossaryheader}{}%
13081 \renewcommand*\{\glsgroupheading}[1]{}%
13082 \renewcommand{\glossentry}[2]{%
13083 \hangindent0pt\relax
13084 \parindent0pt\relax
13085 \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13086 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
13087 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13088 \glstreeprelocation##2\par
13089 }%
13090 \renewcommand{\subglossentry}[3]{%
13091 \hangindent##1\glstreeindent\relax
13092 \parindent##1\glstreeindent\relax
13093 \ifnum##1=1\relax
13094 \glssubentryitem{##2}%
13095 \fi
13096 \glstarget{##2}{\strut}%

```

```

13097      \glossentrydesc{##2}\glspostdescription\glstreechildprelocation##3\par
13098    }%
13099    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13100  }
13101 }
13102 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

13103 \ifdef{\@glsstyle@treenonamegroup}
13104 {%
13105   \renewglossarystyle{treenonamegroup}{%
13106     \setglossarystyle{treenoname}%
13107     \renewcommand{\glsgroupheading}[1]{\par
13108       \noindent\glstreegroupheaderfmt
13109       {\glsggetgrouptitle{##1}}}%
13110     \nopagebreak\indexspace\nobreak\@afterheading
13111   }%
13112 }
13113 }
13114 {}

```

Similarly for `treenonamehypergroup`

```

13115 \ifdef{\@glsstyle@treenonamehypergroup}
13116 {%
13117   \renewglossarystyle{treenonamehypergroup}{%
13118     \setglossarystyle{treenoname}%
13119     \renewcommand*{\glossaryheader}{%
13120       \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13121       \nobreak\@afterheading\indexspace}%
13122     \renewcommand*{\glsgroupheading}[1]{%
13123       \par\noindent
13124       \glstreegroupheaderfmt
13125       {\glsnavhypertarget{##1}{\glsggetgrouptitle{##1}}}%
13126     \nopagebreak\indexspace\nobreak\@afterheading}%
13127   }
13128 }
13129 {}

```

The `almtree` style is redefined to make it easier to made minor adjustments.

```

13130 \ifdef{\@glsstyle@almtree}
13131 {%

```

Only redefine this style if it's already been defined.

mbolDescLocation

```
\glxtralmtreeSymbolDescLocation{\langle label \rangle}{\langle location list \rangle}
```

Layout the symbol, description and location for top-level entries.

```

13132 \newcommand{\glxtralmtreeSymbolDescLocation}[2]{%

```

```

13133   {%
13134       \let\par\glxstrAltTreePar
13135       \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
13136       \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
13137   }%
13138 }

```

trAltTreeIndent Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

13139 \newlength\glxstrAltTreeIndent

```

lsxtrAltTreePar Multi-paragraph descriptions need to keep the hanging indent.

```

13140 \newcommand{\glxstrAltTreePar}{%
13141     @@par
13142     \glxstrAltTreeSetHangIndent
13143     \setlength{\parindent}{\dimexpr\hangindent+\glxstrAltTreeIndent}%
13144 }

```

symbolDescLocation `\glxstralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

13145 \newcommand{\glxstralttreeSubSymbolDescLocation}[3]{%
13146     \glxstralttreeSymbolDescLocation{#2}{#3}%
13147 }

```

trreetopindent 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.

```

13148 \newlength\glxstrtreetopindent

```

sxtralttreeInit User-level initialisation for the alttree style.

```

13149 \newcommand*{\glxstralttreeInit}{%
13150     \settowidth{\glxstrtreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
13151     \glxstrAltTreeIndent=\parindent
13152 }

```

\glsglsetwidest The original `\glsglsetwidest` only uses `\def`. This uses `\gdef`.

```

13153 \newcommand*{\glsglsetwidest}[2][0]{%
13154     \csgdef{@glswidestname\romannumeral#1}{#2}%
13155 }

```

\eglssetwidest The original `\glsglsetwidest` only uses `\def`. This uses `\protected@csdef`.

```

13156 \newcommand*{\eglssetwidest}[2][0]{%
13157     \protected@csdef{@glswidestname\romannumeral#1}{#2}%
13158 }

```

`\xglissetwidest` Like the above but uses `\protected@csxdef`.

```
13159 \newcommand*\xglissetwidest}[2][0]{%
13160   \protected@csxdef{#1}{#2}%
13161 }
```

`glsupdatewidest` Only sets if new value is wider than old value.

```
13162 \newcommand*\glsupdatewidest}[2][0]{%
13163   \ifcsundef{#1}%
13164   {\csdef{#1}{#2}}%
13165   {%
13166     \settowidth{#1}{\csuse{#1}}%
13167     \settowidth{#2}%
13168     \ifdim#1>#2%
13169     \csdef{#1}{#2}%
13170   \fi
13171   }%
13172 }
```

`glsupdatewidest` As above but global definition.

```
13173 \newcommand*\glsupdatewidest}[2][0]{%
13174   \ifcsundef{#1}%
13175   {\csgdef{#1}{#2}}%
13176   {%
13177     \settowidth{#1}{\csuse{#1}}%
13178     \settowidth{#2}%
13179     \ifdim#1>#2%
13180     \csgdef{#1}{#2}%
13181   \fi
13182   }%
13183 }
```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```
13184 \newcommand*\eglsupdatewidest}[2][0]{%
13185   \ifcsundef{#1}%
13186   {\protected@csedef{#1}{#2}}%
13187   {%
13188     \settowidth{#1}{\csuse{#1}}%
13189     \settowidth{#2}%
13190     \ifdim#1>#2%
13191     \protected@csedef{#1}{#2}%
13192   \fi
13193   }%
13194 }
```

`glsupdatewidest` As above but global.

```
13195 \newcommand*\xglsupdatewidest}[2][0]{%
13196   \ifcsundef{#1}%
13197   {\protected@csxdef{#1}{#2}}%
13198   {%
```

```

13199      \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13200      \settowidth{\dimen@ii}{#2}%
13201      \ifdim\dimen@ii>\dimen@
13202          \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13203      \fi
13204  }%
13205  }

```

glswidestname Provide a user-level macro to obtain the widest top-level name.

```

13206  \newcommand*{\glswidestname}{\@glswidestname}

```

glswidestsubname Provide a user-level macro to obtain the widest sub-entry name.

```

13207  \newcommand*{\glswidestsubname}[1]{%
13208      \ifcsundef{@glswidestname\romannumeral#1}%
13209      {\@glswidestname}%
13210      {\csuse{@glswidestname\romannumeral#1}}%
13211  }

```

glswidestTopLevelName CamelCase is easier for long command names. Provide a CamelCase synonym of `\glswidesttoplevelname`.

```

13212  \let\glswidestTopLevelName\glswidesttoplevelname

```

glswidestUsedTopLevelName Like `\glswidesttoplevelname` 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.

```

13213  \newrobustcmd*{\glswidestUsedTopLevelName}[1][\@glo@types]{%
13214      \dimen@=0pt\relax
13215      \gls@tmplen=0pt\relax
13216      \foralllglossaries[#1]{\@gls@type}%
13217      {%
13218          \forglsentries[\@gls@type]{\@glo@label}%
13219          {%
13220              \ifglswidest{\@glo@label}%
13221              {%
13222                  \ifglshasparent{\@glo@label}%
13223                  {}%
13224                  {%
13225                      \settowidth{\dimen@}{%
13226                          {\glstreenamfmt{\glstryname{\@glo@label}}}%
13227                      }%
13228                      \ifdim\dimen@>\gls@tmplen
13229                          \gls@tmplen=\dimen@
13230                      \eglssetwidest{\glstryname{\@glo@label}}%
13231                  }%
13232              }%
13233          }%
13234      }%
13235  }%
13236  }

```

`destUsedAnyName` Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

13237 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
13238 \dimen@=0pt\relax
13239 \gls@tmplen=0pt\relax
13240 \forallglossaries[#1]{\@gls@type}%
13241 {%
13242 \forallglsentries[\@gls@type]{\@glo@label}%
13243 {%
13244 \ifglsused{\@glo@label}%
13245 {%
13246 \settowidth{\dimen@}%
13247 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13248 \ifdim\dimen@>\gls@tmplen
13249 \gls@tmplen=\dimen@
13250 \eglssetwidest{\glsentryname{\@glo@label}}%
13251 \fi
13252 }%
13253 }%
13254 }%
13255 }%
13256 }

```

`ndWidestAnyName` Like the above but doesn't check is the entry has been used.

```

13257 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
13258 \dimen@=0pt\relax
13259 \gls@tmplen=0pt\relax
13260 \forallglossaries[#1]{\@gls@type}%
13261 {%
13262 \forallglsentries[\@gls@type]{\@glo@label}%
13263 {%
13264 \settowidth{\dimen@}%
13265 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13266 \ifdim\dimen@>\gls@tmplen
13267 \gls@tmplen=\dimen@
13268 \eglssetwidest{\glsentryname{\@glo@label}}%
13269 \fi
13270 }%
13271 }%
13272 }

```

`estUsedLevelTwo` This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```

13273 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
13274 \dimen@=0pt\relax
13275 \dimen@i=0pt\relax
13276 \dimen@ii=0pt\relax
13277 \forallglossaries[#1]{\@gls@type}%
13278 {%

```

```

13279 \forglsentries[\@gls@type]{\@glo@label}%
13280 {%
13281 \ifglsused{\@glo@label}%
13282 {%
13283 \ifglsashasparent{\@glo@label}%
13284 {%
13285 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
13286 \ifglsashasparent{\@glo@parent}%
13287 {%
13288 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
13289 \ifglsashasparent{\@glo@parent}%
13290 {}%
13291 {%
13292 \settowidth{\gls@tmplen}%
13293 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13294 \ifdim\gls@tmplen>\dimen@ii
13295 \dimen@ii=\gls@tmplen
13296 \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13297 \fi
13298 }%
13299 }%
13300 {%
13301 \settowidth{\gls@tmplen}%
13302 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13303 \ifdim\gls@tmplen>\dimen@i
13304 \dimen@i=\gls@tmplen
13305 \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13306 \fi
13307 }%
13308 }%
13309 {%
13310 \settowidth{\gls@tmplen}%
13311 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13312 \ifdim\gls@tmplen>\dimen@
13313 \dimen@=\gls@tmplen
13314 \eglssetwidest{\glsentryname{\@glo@label}}%
13315 \fi
13316 }%
13317 }%
13318 {}%
13319 }%
13320 }%
13321 }

```

`dWidestLevelTwo` This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

13322 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
13323 \dimen@=0pt\relax
13324 \dimen@i=0pt\relax
13325 \dimen@ii=0pt\relax

```



```

13326 \forall glossaries[#1]{\@gls@type}%
13327 {%
13328 \forall glsentries[\@gls@type]{\@glo@label}%
13329 {%
13330 \ifgls@hasparent{\@glo@label}%
13331 {%
13332 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
13333 \ifgls@hasparent{\@glo@parent}%
13334 {%
13335 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
13336 \ifgls@hasparent{\@glo@parent}%
13337 {}%
13338 {%
13339 \settowidth{\gls@tmplen}%
13340 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13341 \ifdim\gls@tmplen>\dimen@ii
13342 \dimen@ii=\gls@tmplen
13343 \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13344 \fi
13345 }%
13346 }%
13347 {%
13348 \settowidth{\gls@tmplen}%
13349 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13350 \ifdim\gls@tmplen>\dimen@i
13351 \dimen@i=\gls@tmplen
13352 \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13353 \fi
13354 }%
13355 }%
13356 {%
13357 \settowidth{\gls@tmplen}%
13358 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13359 \ifdim\gls@tmplen>\dimen@
13360 \dimen@=\gls@tmplen
13361 \eglssetwidest{\glsentryname{\@glo@label}}%
13362 \fi
13363 }%
13364 }%
13365 }%
13366 }

```

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.

```

13367 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
13368 \dimen@=0pt\relax
13369 \gls@tmplen=0pt\relax
13370 #2=0pt\relax
13371 \forall glossaries[#1]{\@gls@type}%

```

```

13372  {%
13373    \forglsentries[\@gls@type]{\@glo@label}%
13374  {%
13375    \ifglsused{\@glo@label}%
13376  {%
13377    \settowidth{\dimen@}%
13378      {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13379    \ifdim\dimen@>\gls@tmplen
13380      \gls@tmplen=\dimen@
13381      \eglssetwidest{\glsentryname{\@glo@label}}%
13382    \fi
13383    \settowidth{\dimen@}%
13384      {\glsentrysymbol{\@glo@label}}%
13385    \ifdim\dimen@>#2\relax
13386      #2=\dimen@
13387    \fi
13388  }%
13389  {%
13390  }%
13391  }%
13392  }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

13393  \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
13394    \dimen@=0pt\relax
13395    \gls@tmplen=0pt\relax
13396    #2=0pt\relax
13397    \forallglossaries[#1]{\@gls@type}%
13398  {%
13399    \forglsentries[\@gls@type]{\@glo@label}%
13400  {%
13401    \settowidth{\dimen@}%
13402      {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13403    \ifdim\dimen@>\gls@tmplen
13404      \gls@tmplen=\dimen@
13405      \eglssetwidest{\glsentryname{\@glo@label}}%
13406    \fi
13407    \settowidth{\dimen@}%
13408      {\glsentrysymbol{\@glo@label}}%
13409    \ifdim\dimen@>#2\relax
13410      #2=\dimen@
13411    \fi
13412  }%
13413  }%
13414  }

```

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.

```

13415 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
13416   \dimen@=0pt\relax
13417   \gls@tmplen=0pt\relax
13418   #2=0pt\relax
13419   #3=0pt\relax
13420   \foralllglossaries[#1]{\@gls@type}%
13421   {%
13422     \forglsentries[\@gls@type]{\@glo@label}%
13423     {%
13424       \ifglsused{\@glo@label}%
13425       {%
13426         \settowidth{\dimen@}%
13427           {\glstreenamefmt{\glsentryname{\@glo@label}}}%
13428         \ifdim\dimen@>\gls@tmplen
13429           \gls@tmplen=\dimen@
13430           \eglssetwidest{\glsentryname{\@glo@label}}%
13431         \fi
13432         \settowidth{\dimen@}%
13433           {\glsentrysymbol{\@glo@label}}%
13434         \ifdim\dimen@>#2\relax
13435           #2=\dimen@
13436         \fi
13437         \settowidth{\dimen@}%
13438           {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
13439         \ifdim\dimen@>#3\relax
13440           #3=\dimen@
13441         \fi
13442       }%
13443     }%
13444   }%
13445 }%
13446 }
```

`eSymbolLocation` Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

13447 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
13448   \dimen@=0pt\relax
13449   \gls@tmplen=0pt\relax
13450   #2=0pt\relax
13451   #3=0pt\relax
13452   \foralllglossaries[#1]{\@gls@type}%
13453   {%
13454     \forglsentries[\@gls@type]{\@glo@label}%
13455     {%
13456       \settowidth{\dimen@}%
13457         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
13458       \ifdim\dimen@>\gls@tmplen
13459         \gls@tmplen=\dimen@
13460         \eglssetwidest{\glsentryname{\@glo@label}}%
```

```

13461      \fi
13462      \settowidth{\dimen@}%
13463      {\glstrysymbol{\@glo@label}}}%
13464      \ifdim\dimen@>#2\relax
13465      #2=\dimen@
13466      \fi
13467      \settowidth{\dimen@}%
13468      {\GlsXtrFormatLocationList{\glstrynumberlist{\@glo@label}}}%
13469      \ifdim\dimen@>#3\relax
13470      #3=\dimen@
13471      \fi
13472    }%
13473  }%
13474 }

```

AnyNameLocation Like the `\glstryFindWidestUsedAnyNameSymbolLocation` 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.

```

13475 \newrobustcmd*{\glstryFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
13476   \dimen@=0pt\relax
13477   \glstry@tmplen=0pt\relax
13478   #2=0pt\relax
13479   \forallglossaries[#1]{\@gls@type}%
13480   {%
13481     \forallglsentries[\@gls@type]{\@glo@label}%
13482     {%
13483       \ifglstryused{\@glo@label}%
13484       {%
13485         \settowidth{\dimen@}%
13486         {\glstrynamefmt{\glstryname{\@glo@label}}}%
13487         \ifdim\dimen@>\glstry@tmplen
13488         \glstry@tmplen=\dimen@
13489         \glstrysetwidest{\glstryname{\@glo@label}}%
13490       \fi
13491       \settowidth{\dimen@}%
13492       {\GlsXtrFormatLocationList{\glstrynumberlist{\@glo@label}}}%
13493       \ifdim\dimen@>#2\relax
13494       #2=\dimen@
13495     \fi
13496   }%
13497 }%
13498 }%
13499 }%
13500 }

```

AnyNameLocation Like the `\glstryFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

13501 \newrobustcmd*{\glstryFindWidestAnyNameLocation}[2][\@glo@types]{%
13502   \dimen@=0pt\relax
13503   \glstry@tmplen=0pt\relax

```

```

13504      #2=Opt\relax
13505      \forallglossaries[#1]{\@gls@type}%
13506      {%
13507        \forglsentries[\@gls@type]{\@glo@label}%
13508        {%
13509          \settowidth{\dimen@}%
13510            {\glstreenamefmt{\glsentryname{\@glo@label}}}%
13511          \ifdim\dimen@>\gls@tmplen
13512            \gls@tmplen=\dimen@
13513            \eglssetwidest{\glsentryname{\@glo@label}}%
13514          \fi
13515          \settowidth{\dimen@}%
13516            {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
13517          \ifdim\dimen@>#2\relax
13518            #2=\dimen@
13519          \fi
13520        }%
13521      }%
13522    }

```

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`.

```

13523    \newcommand*{\glsxtrComputeTreeIndent}[1]{%
13524      \glstreeindent=\glsxtrtreetopindent\relax
13525    }

```

computeTreeSubIndent `\glsxtrComputeTreeSubIndent{<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.

```

13526    \newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
13527      \ifcsundef{@glswidestname\romannumeral#1}%
13528      {%
13529        \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
13530      }%
13531      {%
13532        \settowidth{#3}{\glstreenamefmt{%
13533          \csname @glswidestname\romannumeral#1\endcsname\space}}%
13534      }%
13535    }

```

treeSetHangIndent Set `\hangindent` for top-level entries:

```

13536    \newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}

```

etSubHangIndent Set \hangindent for sub-entries:

```
13537 \newcommand*{\glxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
13538 \renewglossarystyle{alttree}{%
13539   \renewenvironment{theglossary}%
13540     {%
13541       \glxtralttreeInit
13542       \def\@gls@prevlevel{-1}%
13543       \mbox{}\par}%
13544     {\par}%
13545   \renewcommand*{\glossaryheader}{}%
13546   \renewcommand*{\glsgroupheading}[1]{}%
13547   \renewcommand{\glossentry}[2]{%
13548     \ifnum\@gls@prevlevel=0\relax
13549     \else
13550       \glxtrComputeTreeIndent{##1}%
13551     \fi
13552     \parindent\glstreeindent
13553     \glxtrAltTreeSetHangIndent
13554     \makebox[0pt][r]%
13555     {%
13556       \glstreenamebox{\glstreeindent}%
13557       {%
13558         \glstryitem{##1}%
13559         \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13560       }%
13561     }%
13562     \glxtralttreeSymbolDescLocation{##1}{##2}%
13563     \def\@gls@prevlevel{0}%
13564   }
13565   \renewcommand{\subglossentry}[3]{%
13566     \ifnum##1=1\relax
13567       \glssubentryitem{##2}%
13568     \fi
13569     \ifnum\@gls@prevlevel=##1\relax
13570     \else
13571       \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
13572       \ifnum\@gls@prevlevel<##1\relax
13573         \setlength\glstreeindent\gls@tmplen
13574         \addtolength\glstreeindent\parindent
13575         \parindent\glstreeindent
13576       \else
13577         \ifnum\@gls@prevlevel=0\relax
13578           \glxtrComputeTreeIndent{##2}%
13579         \else
13580           \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
13581         \fi
13582         \addtolength\parindent{-\glstreeindent}%

```

```

13583         \setlength\glstreeindent\parindent
13584     \fi
13585 \fi
13586 \glstrAltTreeSetSubHangIndent{##1}%
13587 \makebox[Opt][r]{\glstreenamebox{\gls@tmplen}{%
13588     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
13589 \glstralttreeSubSymbolDescLocation{##1}{##2}{##3}%
13590 \def\@gls@prevlevel{##1}%
13591 }%
13592 \renewcommand*\{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13593 }
13594 }%
13595 {%
13596 }

```

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.

```

13597 \ifdef{\@glsstyle@alttreegroup}
13598 {%
13599     \renewglossarystyle{alttreegroup}{%
13600         \setglossarystyle{alttree}%
13601         \renewcommand{\glsgroupheading}[1]{\par
13602             \def\@gls@prevlevel{-1}%
13603             \hangindentOpt\relax
13604             \parindentOpt\relax
13605             \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
13606         \nopagebreak\indexspace\nopagebreak
13607     }%
13608 }%
13609 }%
13610 {%
13611 }

```

Similarly for `alttreehypergroup`.

```

13612 \ifdef{\@glsstyle@alttreehypergroup}
13613 {%
13614     \renewglossarystyle{alttreehypergroup}{%
13615         \setglossarystyle{alttree}%
13616         \renewcommand*\{\glossaryheader}{%
13617             \par
13618             \def\@gls@prevlevel{-1}%
13619             \hangindentOpt\relax
13620             \parindentOpt\relax
13621             \glstreenavigationfmt{\glsnavigation}\par\indexspace
13622         }%
13623         \renewcommand*\{\glsgroupheading}[1]{%
13624             \par
13625             \def\@gls@prevlevel{-1}%
13626             \hangindentOpt\relax
13627             \parindentOpt\relax

```

```

13628     \glstreegroupheaderfmt
13629     {\glsnabhypertarget{##1}{\glsgrouptitle{##1}}}\par
13630     \nopagebreak\indexspace\nopagebreak
13631 }%
13632 }
13633 }%
13634 {%
13635 }

```

2.9 Multicolumn Styles

Adjust mcolindexgroup to discourage page breaks after the group headings.

```

13636 \ifdef{\@glsstyle@mcolindexgroup}
13637 {%
13638   \renewglossarystyle{mcolindexgroup}{%
13639     \setglossarystyle{mcolindex}%
13640     \renewcommand*\{glsgroupheading}[1]{%
13641       \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
13642       \nopagebreak\indexspace\nobreak\@afterheading
13643     }%
13644   }
13645 }%
13646 {%
13647 }

```

Similarly for mcolindexhypergroup.

```

13648 \ifdef{\@glsstyle@mcolindexhypergroup}
13649 {%
13650   \renewglossarystyle{mcolindexhypergroup}{%
13651     \setglossarystyle{mcolindex}%
13652     \renewcommand*\{glossaryheader}{%
13653       \item\glstreenavigationfmt{\glsnavigation}%
13654       \indexspace
13655     }%
13656     \renewcommand*\{glsgroupheading}[1]{%
13657       \item\glstreegroupheaderfmt
13658         {\glsnabhypertarget{##1}{\glsgrouptitle{##1}}}%
13659       \nopagebreak\indexspace\nobreak\@afterheading
13660     }%
13661   }
13662 }%
13663 {%
13664 }

```

Similarly for mcolindexspannav.

```

13665 \ifdef{\@glsstyle@mcolindexspannav}
13666 {%
13667   \renewglossarystyle{mcolindexspannav}{%
13668     \setglossarystyle{index}%

```



```

13669 \renewenvironment{theglossary}%
13670 {%
13671 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]]%
13672 \setlength{\parindent}{0pt}%
13673 \setlength{\parskip}{0pt plus 0.3pt}%
13674 \let\item\glstreeitem}%
13675 {\end{multicols}}}%
13676 \renewcommand*{\glsgroupheading}[1]{%
13677 \item\glstreegroupheaderfmt
13678 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13679 \nopagebreak\indexspace\nobreak\@afterheading
13680 }%
13681 }
13682 }%
13683 {%
13684 }

```

Similarly for mcoltreegroup.

```

13685 \ifdef{\@glstyle@mcoltreegroup}
13686 {%
13687 \renewglossarystyle{mcoltreegroup}{%
13688 \setglossarystyle{mcoltree}%
13689 \renewcommand{\glsgroupheading}[1]{\par
13690 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}}%
13691 \nopagebreak\indexspace\nobreak\@afterheading
13692 }%
13693 }
13694 }%
13695 {%
13696 }

```

Similarly for mcoltreehypergroup.

```

13697 \ifdef{\@glstyle@mcoltreehypergroup}
13698 {%
13699 \renewglossarystyle{mcoltreehypergroup}{%
13700 \setglossarystyle{mcoltree}%
13701 \renewcommand*{\glossaryheader}{%
13702 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
13703 }%
13704 \renewcommand*{\glsgroupheading}[1]{%
13705 \par\noindent
13706 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13707 \nopagebreak\indexspace\nobreak\@afterheading
13708 }%
13709 }
13710 }%
13711 {%
13712 }

```

Similarly for mcoltreespannav.

```

13713 \ifdef{\@glstyle@mcoltreespannav}

```

```

13714 {%
13715   \renewglossarystyle{mcoltreespannav}{%
13716     \setglossarystyle{tree}%
13717     \renewenvironment{theglossary}%
13718     {%
13719       \begin{multicols}{\glsmcols}%
13720       [\noindent\glstreenavigationfmt{\glsnavigation}]%
13721       \setlength{\parindent}{0pt}%
13722       \setlength{\parskip}{0pt plus 0.3pt}%
13723     }%
13724     {\end{multicols}}}%
13725   \renewcommand*{\glsgroupheading}[1]{%
13726     \par\noindent
13727     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13728     \nopagebreak\indexspace\nobreak\@afterheading
13729   }%
13730 }
13731 }%
13732 {%
13733 }

```

Similarly for mcoltreenonamegroup.

```

13734 \ifdef{\@glsstyle@mcoltreenonamegroup}
13735 {%
13736   \renewglossarystyle{mcoltreenonamegroup}{%
13737     \setglossarystyle{mcoltreenoname}%
13738     \renewcommand*{\glsgroupheading}[1]{\par
13739       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
13740     \nopagebreak\indexspace\nobreak\@afterheading
13741   }%
13742 }
13743 }%
13744 {%
13745 }

```

Similarly for mcoltreenonamehypergroup.

```

13746 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
13747 {%
13748   \renewglossarystyle{mcoltreenonamehypergroup}{%
13749     \setglossarystyle{mcoltreenoname}%
13750     \renewcommand*{\glossaryheader}{%
13751       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
13752     \renewcommand*{\glsgroupheading}[1]{%
13753       \par\noindent
13754       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13755       \nopagebreak\indexspace\nobreak\@afterheading}%
13756   }
13757 }%
13758 {%
13759 }

```

Similarly for mcoltreenonamespannav.

```

13760 \ifdef{\@glsstyle@mcoltreenonamespannav}
13761 {%
13762   \renewglossarystyle{mcoltreenonamespannav}{%
13763     \setglossarystyle{treenoname}%
13764     \renewenvironment{theglossary}%
13765     {%
13766       \begin{multicols}{\glsmcols}%
13767       [\noindent\glstreenavigationfmt{\glsnavigation}}%
13768       \setlength{\parindent}{0pt}%
13769       \setlength{\parskip}{0pt plus 0.3pt}%
13770     }%
13771     {\end{multicols}}}%
13772   \renewcommand*{\glsgroupheading}[1]{%
13773     \par\noindent
13774     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13775     \nopagebreak\indexspace\nobreak\@afterheading}%
13776   }
13777 }%
13778 {%
13779 }

```

mcolalmtree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{} \par which would unbalance the top of the columns.

```

13780 \ifdef{\@glsstyle@mcolalmtree}
13781 {%
13782   \renewglossarystyle{mcolalmtree}{%
13783     \setglossarystyle{almtree}%
13784     \renewenvironment{theglossary}%
13785     {%
13786       \glxtralttreeInit
13787       \def\@gls@prevlevel{-1}%
13788       \begin{multicols}{\glsmcols}%
13789     }%
13790     {\par\end{multicols}}}%
13791   }
13792 }%
13793 {%
13794 }

```

Redefine mcolalmtreegroup to discourage page breaks after the group headings.

```

13795 \ifdef{\@glsstyle@mcolalmtreegroup}
13796 {%
13797   \renewglossarystyle{mcolalmtreegroup}{%
13798     \setglossarystyle{mcolalmtree}%
13799     \renewcommand{\glsgroupheading}[1]{\par
13800       \def\@gls@prevlevel{-1}%
13801       \hangindent0pt\relax
13802       \parindent0pt\relax
13803       \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%

```

```

13804     \nopagebreak\indexspace\nopagebreak
13805 }%
13806 }
13807 }%
13808 {%
13809 }

```

Similarly for mcolalmtreehypergroup.

```

13810 \ifdef{\@glsstyle@mcolalmtreehypergroup}
13811 {%
13812   \renewglossarystyle{mcolalmtreehypergroup}{%
13813     \setglossarystyle{mcolalmtree}%
13814     \renewcommand*{\glossaryheader}{%
13815       \par
13816       \def\@gls@prevlevel{-1}%
13817       \hangindent0pt\relax
13818       \parindent0pt\relax
13819       \glstreenavigationfmt{\glsnavigation}%
13820       \par\indexspace
13821     }%
13822     \renewcommand*{\glsgroupheading}[1]{%
13823       \par
13824       \def\@gls@prevlevel{-1}%
13825       \hangindent0pt\relax
13826       \parindent0pt\relax
13827       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13828       \nopagebreak\indexspace\nopagebreak
13829     }%
13830   }
13831 }%
13832 {%
13833 }

```

Similarly for mcolalttreespannav.

```

13834 \ifdef{\@glsstyle@mcolalttreespannav}
13835 {%
13836   \renewglossarystyle{mcolalttreespannav}{%
13837     \setglossarystyle{alttree}%
13838     \renewenvironment{theglossary}%
13839     {%
13840       \glsextralmtreeInit
13841       \def\@gls@prevlevel{-1}%
13842       \begin{multicols}{\glsmcols}%
13843         [\noindent\glstreenavigationfmt{\glsnavigation}]%
13844       }%
13845       {\par\end{multicols}}%
13846     \renewcommand*{\glsgroupheading}[1]{%
13847       \par
13848       \def\@gls@prevlevel{-1}%
13849       \hangindent0pt\relax

```

```

13850      \parindent0pt\relax
13851      \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgrouptitle{##1}}}%
13852      \nopagebreak\indexspace\nopagebreak
13853    }%
13854  }
13855}%
13856{%
13857}

Reset the default style
13858\ifx\@glossary@default@style\relax
13859\else
13860  \setglossarystyle{\@glxtr@current@style}
13861\fi

```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
13862 \NeedsTeXFormat{LaTeX2e}
13863 \ProvidesPackage{glossary-bookindex}[2018/02/26 v1.27 (NLCT)]
```

Load required packages.

```
13864 \RequirePackage{multicol}
13865 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
13866 \newcommand{\glstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
13867 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

`ookindexsubname` Format used for sub entries.

```
13868 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

`strprelocation` Provide in case glossaries-stylemods isn't loaded.

```
13869 \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.

```
13870 \newcommand*{\glstrbookindexprelocation}[1]{%
13871   \glstrifhasfield{location}{#1}%
13872   {,\glstrprelocation}%
13873   {\glstrprelocation}%
13874 }
```

`xsubprelocation` Separator used before location list for sub-entries.

```
13875 \newcommand*{\glstrbookindexsubprelocation}[1]{%
13876   \glstrbookindexprelocation{#1}%
13877 }
```

`xparentchildsep` Separator used between top-level parent and child entry.

```
13878 \newcommand{\glstrbookindexparentchildsep}{\nopagebreak}
```

`rentsubchildsep` Separator used between sub-level parent and child entry.

```
13879 \newcommand{\glstrbookindexparentsubchildsep}{\glstrbookindexparentchildsep}
```

`bookindexbetween` Between two top-level entries identified by the labels in the arguments.

```
13880 \newcommand{\glstrbookindexbetween}[2]{}

indexsubbetween Between two level 1 entries identified by the labels in the arguments.
13881 \newcommand{\glstrbookindexsubbetween}[2]{}

exsubsubbetween Between two level 2 entries identified by the labels in the arguments.
13882 \newcommand{\glstrbookindexsubsubbetween}[2]{}

indexatendgroup At the end of a letter group. The argument is the index of the last top-level entry.
13883 \newcommand{\glstrbookindexatendgroup}[1]{}

exsubatendgroup At the end of a letter group. The argument is the index of the last level 1 entry.
13884 \newcommand{\glstrbookindexsubatendgroup}[1]{}

subsubatendgroup At the end of a letter group. The argument is the index of the last level 2 entry.
13885 \newcommand{\glstrbookindexsubsubatendgroup}[1]{}

kindexgroupskip Group separator.
13886 \newcommand{\glstrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}

Format group title.

dexformatheader Group separator.
13887 \newcommand*{\glstrbookindexformatheader}[1]{%
13888 \par{\centering\glstreegroupheaderfmt{#1}\par}%
13889 }

okindexbookmark Book mark group heading if supported.
13890 \ifdef\pdfbookmark
13891 {%
13892 \newcommand*{\glstrbookindexbookmark}[2]{%
13893 \ifdefstring{\@@glossarysec}{chapter}%
13894 {\pdfbookmark[1]{#1}{#2}}%
13895 {\pdfbookmark[2]{#1}{#2}}%
13896 }
13897 }
13898 {%
13899 \newcommand*{\glstrbookindexbookmark}[2]{}
13900 }

kindexcolspread
13901 \newcommand*{\glstrbookindexcolspread}{}

dexmulticolenv
13902 \newcommand*{\glstrbookindexmulticolenv}{multicols}
```

Define the style.

```

13903 \newglossarystyle{bookindex}{%
13904   \setglossarystyle{index}%
13905   \renewenvironment{theglossary}%
13906   {%
13907     \ifdefempty{glxstrbookindexcolspread}
13908     {%
13909       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
13910       {\glxstrbookindexcols}%
13911     }%
13912     {%
13913       \expandafter\begin\expandafter{\glxstrbookindexmulticolseenv}%
13914       {\glxstrbookindexcols}[\glxstrbookindexcolspread]%
13915     }%
13916     \setlength{\parindent}{0pt}%
13917     \setlength{\parskip}{0pt plus 0.3pt}%
13918     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
13919     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
13920     \let\@glxstr@bookindex@between\@gobble
13921     \let\@glxstr@bookindex@subbetween\@gobble
13922     \let\@glxstr@bookindex@subsubbetween\@gobble
13923     \let\@glxstr@bookindex@atendgroup\relax
13924     \let\@glxstr@bookindex@subatendgroup\relax
13925     \let\@glxstr@bookindex@subsubatendgroup\relax
13926     \let\@glxstr@bookindex@groupskip\relax
13927   }%
13928   {%

```

Do end group hooks.

```

13929     \@glxstr@bookindex@subsubatendgroup
13930     \@glxstr@bookindex@subatendgroup
13931     \@glxstr@bookindex@atendgroup

```

End multicol environment.

```

13932     \expandafter\end\expandafter{\glxstrbookindexmulticolseenv}%
13933   }%

```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```

13934   \renewcommand*{\glossaryheader}{\raggedright}%

```

Top level entry format.

```

13935   \renewcommand*{\glossentry}[2]{%

```

Do separator.

```

13936     \@glxstr@bookindex@between{##1}%

```

Update separators.

```

13937     \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
13938     \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
13939     \let\@glxstr@bookindex@subbetween\@gobble
13940     \let\@glxstr@bookindex@subsubbetween\@gobble
13941     \edef\@glxstr@bookindex@between{%

```



```

13942     \noexpand\glxstrbookindexbetween{##1}%
13943 }%
13944 \edef\@glxstr@bookindex@atendgroup{%
13945     \noexpand\glxstrbookindexatendgroup{##1}%
13946 }%
13947 \let\@glxstr@bookindex@subatendgroup\relax
13948 \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

13949 \glstreeitem
13950 \glstryitem{##1}%
13951 \glstarget{##1}{\glxstrbookindexname{##1}}%
13952 \glxstrbookindexprelocation{##1}##2%
13953 }%
13954 \renewcommand{\subglossentry}[3]{%
13955     \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```

13956 \glstreeitem
13957 \or

```

Level 1.

```

13958 \@glxstr@bookindex@sep
13959 \@glxstr@bookindex@subbetween{##2}%
13960 \let\@glxstr@bookindex@sep\relax

```

Update separators.

```

13961 \let\@glxstr@bookindex@subsubbetween\@gobble
13962 \let\@glxstr@bookindex@subsep\glxstrbookindexparentsubchildsep
13963 \edef\@glxstr@bookindex@subbetween{%
13964     \noexpand\glxstrbookindexsubbetween{##2}%
13965 }%
13966 \edef\@glxstr@bookindex@atsubendgroup{%
13967     \noexpand\glxstrbookindexatsubendgroup{##1}%
13968 }%

```

Start sub-item.

```

13969 \glstreesubitem
13970 \glssubentryitem{##2}%
13971 \else

```

All other levels.

```

13972 \@glxstr@bookindex@subsep
13973 \@glxstr@bookindex@subsubbetween{##2}%

```

Update separators.

```

13974 \let\@glxstr@bookindex@subsep\relax
13975 \edef\@glxstr@bookindex@subsubbetween{%
13976     \noexpand\glxstrbookindexsubsubbetween{##2}%
13977 }%
13978 \edef\@glxstr@bookindex@atsubsubendgroup{%
13979     \noexpand\glxstrbookindexatsubsubendgroup{##1}%
13980 }%

```

Start sub-sub-item.

```
13981 \glstreesubsubitem
13982 \fi
```

Format entry.

```
13983 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
13984 \glxtrbookindexsubprelocation{##2}##3%
13985 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
13986 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
13987 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
13988 \@glxtr@bookindex@subsubatendgroup
13989 \@glxtr@bookindex@subatendgroup
13990 \@glxtr@bookindex@atendgroup
13991 \@glxtr@bookindex@groupskip
```

Update separators.

```
13992 \let\@glxtr@bookindex@groupskip\glxtrbookindexgroupskip
13993 \let\@glxtr@bookindex@between\@gobble
13994 \let\@glxtr@bookindex@atendgroup\relax
13995 \let\@glxtr@bookindex@subatendgroup\relax
13996 \let\@glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
13997 \glxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
13998 \glxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
13999 \glxtrbookindexformatheader{\thisgrptitle}%
14000 \nopagebreak\indexspace\nopagebreak\@afterheading
14001 }%
14002 }
```

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.

```
14003 \newcommand{\glxtrbookindexthepage}{%
14004 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
14005 }
```

bookindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
14006 \newcommand*{\glxtrbookindexmarkentry}[1]{%
14007   \protected@write\@auxout
14008   {\let\glxtrbookindexthepage\relax}%
14009   {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
14010 }
```

etbookindexmark

```
14011 \newcommand*{\glxtr@setbookindexmark}[2]{%
14012   \ifcsundef{glxtr@idxfirstmark@#1}%
14013   {\csgdef{glxtr@idxfirstmark@#1}{#2}}%
14014   {}%
14015   \csgdef{glxtr@idxlastmark@#1}{#2}%
14016 }
```

indexfirstmarkfmt

```
14017 \newcommand*{\glxtrbookindexfirstmarkfmt}[1]{%
14018   \glsentryname{#1}%
14019 }
```

indexfirstmark

```
14020 \newcommand*{\glxtrbookindexfirstmark}{%
14021   \letcs{\glxtr@label}{glxtr@idxfirstmark@\glxtrbookindexthepage}%
14022   \ifdef\glxtr@label
14023   {\glxtrbookindexfirstmarkfmt{\glxtr@label}}%
14024   {}%
14025 }
```

indexlastmarkfmt

```
14026 \newcommand*{\glxtrbookindexlastmarkfmt}[1]{%
14027   \glsentryname{#1}%
14028 }
```

okindexlastmark

```
14029 \newcommand*{\glxtrbookindexlastmark}{%
14030   \letcs{\glxtr@label}{glxtr@idxlastmark@\glxtrbookindexthepage}%
14031   \ifdef\glxtr@label
14032   {\glxtrbookindexlastmarkfmt{\glxtr@label}}%
14033   {}%
14034 }
```

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	65
General: Initial experimental release	5	\@Glsymbolplural@: added	
0.2 (2015-11-30)		redefinition	66
\Glsfmtshort: new	306	\@Glstext@: added redefinition	61
\glsfmtshort: new	306	\@Glsuseri@: added redefinition	66
\Glsfmtshortpl: new	306	\@Glsuserii@: added redefinition	66
\glsfmtshortpl: new	306	\@Glsuseriii@: added redefinition	67
short: switched inline full form to short		\@Glsuseriv@: added redefinition	67
(long)	209	\@Glsuserv@: added redefinition	67
0.3 (2015-12-02)		\@Glsuservi@: added redefinition	68
\@ACRlong: added redefinition	71	\@acrlong: added redefinition	70
\@ACRlongpl: added redefinition	72	\@acrlongpl: added redefinition	71
\@ACRshort: added redefinition	69	\@acrshort: added redefinition	68
\@ACRshortpl: added redefinition	70	\@acrshortpl: added redefinition	69
\@Acrlong: added redefinition	70	\@gls@field@link: added optional	
\@Acrlongpl: added redefinition	71	argument	54
\@Acrshort: added redefinition	68	\@glsdescplural@: added redefinition	64
\@Acrshortpl: added redefinition	69	\@glsfirst@: added redefinition	61
\@GLSdesc@: added redefinition	64	\@glsfirstplural@: added redefinition	63
\@GLSdescplural@: added redefinition	65	\@glsplural@: added redefinition	62
\@GLSfirst@: added redefinition	62	\@glssymbolplural@: added	
\@GLSfirstplural@: added redefinition	63	redefinition	65
\@GLSname@: added redefinition	64	\@glsxtr@defaultnoglossarywarning:	
\@GLSplural@: added redefinition	63	new	121
\@GLSSymbol@: added redefinition	65	\@glsxtr@field@linkdefs: new	60
\@GLSSymbolplural@: added		\@glsxtr@insertdots: new	179
redefinition	66	\@print@glossary: added redefinition	118
\@GLStext@: added redefinition	61	\glsabbrvdefaultfont: renamed from	
\@GLSuseri@: added redefinition	66	\abbrvdefaultfont	184
\@GLSuserii@: added redefinition	67	\glsaccessdesc: new	146
\@GLSuseriii@: added redefinition	67	\glsaccessdescplural: new	147
\@GLSuseriv@: added redefinition	67	\glsaccessfirst: new	144
\@GLSuserv@: added redefinition	67	\glsaccessfirstplural: new	144
\@GLSuservi@: added redefinition	68	\Glsaccesslong: new	149
\@GLSdesc@: added redefinition	64	\glsaccesslong: new	148
\@GLSdescplural@: added redefinition	65	\glsaccessname: new	142
\@GLSfirst@: added redefinition	62	\glsaccessplural: new	143
\@GLSfirstplural@: added redefinition	63	\Glsaccessshort: new	148
\@GLSname@: added redefinition	64	\glsaccessshort: new	147
\@GLSplural@: added redefinition	62	\Glsaccessshortpl: new	148

\glsaccesssshortpl: new	148	\cGLSpl: new	95
\glsaccessssymbol: new	145	\cGLSpl@: new	96
\glsaccessssymbolplural: new	146	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	143	new	91
\glsentryfmt: added check for short ..	54	\cGLS: new	95
\glslongpltok: new	179	\cGLSformat: new	95
\glsshortpltok: new	178	\cGLSpl: new	95
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	96
name in \setkeys	180	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	15
for plural	175	\glsenableentrycount: new	91
\GLSxtrlongpl: new	193	\glsfirstabbrvdefaultfont: new ..	184
\Glsxtrlongpl: new	193	\glsfirstlongdefaultfont: new ...	184
\glsxtrlongpl: new	192	\Glsfmtfirst: new	309
\glsxtrNoGlossaryWarning: new	20	\glsfmtfirst: new	308
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	309
new	175	\glsfmtfirstpl: new	309
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	308
new	175	\glsfmtplural: new	308
\glsxtrpostlinkendsentence: new ..	175	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	192	\Glsxtrtitleshort	306
\Glsxtrshortpl: new	191	renamed from \Glsentryfmtshort ..	306
\glsxtrshortpl: new	191	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort	306
\glslabeltok	204	renamed from \glsentryfmtshort ..	306
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	202	\Glsxtrtitleshortpl	306
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glsentryfmtshortpl	306
redefinition of \acronymtype	17	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl	306
\Glsxtrshort	306	renamed from	
\glsfmtshort: changed to use		\glsentryfmtshortpl	306
\glsxtrshort	306	\Glsfmttext: new	308
\Glsfmtshortpl: changed to use		\glsfmttext: new	307
\glsxtrshortpl	306	\glshasattribute: new	154
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	153
\glsxtrshortpl	306	\glsxtremsuffix: new	245
\glsxtrifemptyglossary: new	25	\GlsXtrEnableEntryCounting: new ..	90
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	93
argument	157	\glsxtrscfont: new	217
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	217
argument	157	\glsxtrsmfont: new	231
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	231
default type to \acronymtype	104	short-em: new	252
\newterm: fixed name argument	156	short-em-desc: new	254
0.5 (2015-12-07)		short-em-footnote: new	263
\cGLS: new	95	short-em-long: new	249
\cGLS@: new	95	short-em-long-desc: new	250

short-em-postfootnote: new	265	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	227	attribute	297
short-sc-postfootnote: new	229	\Glsxtrheadtext: now uses headuc	
short-sm: new	235	attribute	299
short-sm-desc: new	236	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	242	attribute	299
short-sm-long: new	233	short-em-footnote: switch off regular	
short-sm-long-desc: new	234	attribute if set	263
short-sm-postfootnote: new	243	short-long: switch off regular attribute	
long-noshort-em: new	256	if set	203
long-noshort-em-desc: new	260	short-long-desc: switch off regular	
long-noshort-sm: new	238	attribute if set	204
long-noshort-sm-desc: new	240	short-sc-footnote: switch off regular	
long-short-em: new	246	attribute if set	228
long-short-em-desc: new	247	short-sm-footnote: switch off regular	
long-short-sm: new	232	attribute if set	242
long-short-sm-desc: new	233	long-short: switch off regular attribute	
0.5.1 (2015-12-02)		if set	201
\Glsaccessstext: new	143	long-short-desc: switch off regular	
0.5.1 (2015-12-07)		attribute if set	202
\@glxtr@doaccsupp: new	20	long-short-sc-desc: switch off regular	
General: removed \ifglxtruseuchead	296	attribute if set	219
\Glsaccessdesc: new	146	footnote: switch off regular attribute if	
\Glsaccessdescplural: new	147	set	205
\Glsaccessfirst: new	144	postfootnote: switch off regular	
\Glsaccessfirstplural: new	145	attribute if set	207
\Glsaccessname: new	142	0.5.2 (2015-12-08)	
\Glsaccessplural: new	143	\@GLSdesc@: added accessibility support	64
\Glsaccesssymbol: new	145	\@GLSdescplural@: added accessibility	
\Glsaccesssymbolplural: new	146	support	65
\glxtrheadfirst: now uses headuc		\@GLSfirst@: added accessibility	
attribute	301	support	62
\glxtrheadfirst: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	301	support	63
\Glsxtrheadfirstplural: now uses		\@GLSname@: added accessibility support	64
headuc attribute	302	\@GLSplural@: added accessibility	
\glxtrheadfirstplural: now uses		support	63
headuc attribute	301	\@GLSsymbol@: added accessibility	
\Glsxtrheadplural: now uses headuc		support	65
attribute	300	\@GLSsymbolplural@: added	
\glxtrheadplural: now uses headuc		accessibility support	66
attribute	300	\@GLStext@: added accessibility support	61
\Glsxtrheadshort: now uses headuc		\@GLSdesc@: added accessibility support	64
attribute	297	\@GLSdescplural@: added accessibility	
\glxtrheadshort: now uses headuc		support	65
attribute	296	\@GLSfirst@: added accessibility	
\Glsxtrheadshortpl: now uses headuc		support	62
attribute	298	\@GLSfirstplural@: added accessibility	
		support	63

\@Glsname@: add accessibility support ..	64	\GLSaccesssymbolplural: new ..	146, 151
\@Glsplural@: added accessibility support	62	\GLSaccessstext: new	143, 150
\@Glsymbol@: added accessibility support	65	\glsetentryfmt: moved	
\@Glsymbolplural@: added accessibility support	66	\glsetabbrvfmt from	
\@Glstext@: added accessibility support	61	\glxtrabbrvfmt to here	54
\@glsgldesc@: added accessibility support	64	\GlsXtrEnableInitialTagging: new	171
\@glsgldescplural@: added accessibility support	64	\glxtrfieldtitlecase: new	158
\@glsglfirst@: added accessibility support	61	\GlsXtrFormatLocationList: new ...	52
\@glsglfirstplural@: added accessibility support	63	\glxtrnewabbrevpresetkeyhook:	
\@glsglname@: added accessibility support	64	new	182
\@glsglplural@: added accessibility support	62	\glxtrtagfont: new	173
\@glsglsymbol@: added accessibility support	65	\KV@printgloss@nonumberlist: added	53
\@glsglsymbolplural@: added accessibility support	65	\mfu@checkword@do: added	172
\@glsglstext@: added accessibility support	61	\setabbreviationstyle: added check	
\@glsglxtr@activate@initialtagging:		for post-definition style switch	197
new	173	0.5.3 (2015-12-09)	
\@glsglxtr@do@titlecaps@warn: new ..	173	\@glsglxtr@autoindex@at: new	169
\@glsglxtr@tag: new	173	\@glsglxtr@autoindex@encap: new ...	169
General: fixed typo in glossaries-accsupp		\@glsglxtr@autoindex@esc: new	170
and tidied up code to use just one		\@glsglxtr@autoindex@level: new ...	169
\@ifpackageloaded	142	\@glsglxtr@autoindex@setname: new ..	167
removed \glxtrabbrvfmt	194	\@glsglxtr@doabbreviationsdef: new ..	16
\glossaryentrynumbers: added	51	General: removed	
\Glossentrydesc: added	171	\GlsXtrNoGlsWarningNoAutoMakeMain	
\Glossentryname: added	163	120
\Glossentrysymbol: added	171	\glsglwidth: added	51
\GLSaccessdesc: new	147, 151	\glspagelistwidth: added	51
\GLSaccessdescplural: new ...	147, 152	\glsglxtrdoautoindexname: new	167
\GLSaccessfirst: new	144, 150	\glsglxtrpostnamehook: new	164
\GLSaccessfirstplural: new ..	145, 151	\if@glsglxtr@format@override: new ..	166
\GLSaccesslong: new	149, 152	\ProvidesGlossariesExtraLang: new	312
\GLSaccesslongpl: new	149, 152	\RequireGlossariesExtraLang: new	312
\Glsaccesslongpl: new	149	0.5.4 (2015-12-15)	
\glsglaccesslongpl: new	149	\@newglossaryentry@defunitcounters:	
\GLSaccessname: new	143, 150	new	96
\GLSaccessplural: new	144, 150	\@GLSxtr@p@acrlong@: new	84
\GLSaccesssshort: new	148, 152	\@GLSxtr@p@acrlongpl@: new	84
\GLSaccesssshortpl: new	148, 152	\@GLSxtr@p@acrshort@: new	83
\GLSaccessssymbol: new	145, 151	\@GLSxtr@p@acrshortpl@: new	83
		\@GLSxtr@p@long@: new	83
		\@GLSxtr@p@longpl@: new	83
		\@GLSxtr@p@plural@: new	82
		\@GLSxtr@p@short@: new	82
		\@GLSxtr@p@shortpl@: new	82
		\@GLSxtr@p@text@: new	81
		\@GlsXtrEnableOnTheFly: new	47
		\@Glsxtr: new	48
		\@Glsxtr@p@acrlong@: new	84
		\@Glsxtr@p@acrlongpl@: new	84

\@Glsxtr@p@acrshort@: new	83	\glsenableentryunitcount: new	99
\@Glsxtr@p@acrshortpl@: new	83	\glshasattribute: added check for	
\@Glsxtr@p@long@: new	83	entry's existence	154
\@Glsxtr@p@longpl@: new	83	\glusifattribute: added check for	
\@Glsxtr@p@plural@: new	82	entry's existence	154
\@Glsxtr@p@short@: new	82	\glspostlinkhook: added existence	
\@Glsxtr@p@shortpl@: new	82	check	174
\@Glsxtr@p@text@: new	81	\Glsxtr: new	48
\@Glsxtrpl: new	49	\glxtr: new	47
\@alt@glshyp@opt: new	78	\glxtrcat: new	47
\@glscalt@hyp@opt: new	77	\glxtrdowrglossaryhook: new	77
\@glscalt@hyp@opt@char: new	78	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new	78	new	102
\@glscincrement@currunitcount:		\GlsXtrEnableOnTheFly: new	46
new	97	\Glsxtrpl: new	48
\@glsclocal@increment@currunitcount:		\glxtrpl: new	48
new	98	\glxtrpostlocalreset: new	90
\@glscsetdefault@glslink@opts:		\glxtrpostlocalunset: new	90
new	75	\glxtrpostreset: new	90
\@glsxtr: new	47	\glxtrpostunset: new	89
\@glsxtr@addunitcounter: new	97	\glxtrprotectlinks: new	81
\@glsxtr@currunitcount: new	98	\GlsXtrSetAltModifier: new	78
\@glsxtr@ifunitcounter: new	97	\GlsXtrSetDefaultGlsOpts: new	76
\@glsxtr@p@acrlong@: new	83	\glxtrstarflywarn: new	47
\@glsxtr@p@acrlongpl@: new	84	\GlsXtrWarning: new	49
\@glsxtr@p@acrshort@: new	83	\MakeAcronymsAbbreviations: now	
\@glsxtr@p@acrshortpl@: new	83	disables \setacronymstyle	104
\@glsxtr@p@long@: new	83	1.0 (2016-01-24)	
\@glsxtr@p@longpl@: new	83	\@glsxtr@autoindexcrossrefs: new	15
\@glsxtr@p@plural@: new	81	\@glsxtr@idx@displaynumberlist:	
\@glsxtr@p@short@: new	82	new	112
\@glsxtr@p@shortpl@: new	82	\@glsxtr@idx@entrynumberlist: new	114
\@glsxtr@p@text@: new	81	\@glsxtr@noidx@displaynumberlist:	
\@glsxtr@prevunitcount: new	98	new	112
\@glsxtr@setentryunitcountunsetattr:		\@glsxtr@noidx@entrynumberlist:	
new	102	new	113
\@glsxtr@unitcountlist: new	97	\@glsxtr@noidx@numberlistloop:	
\@glsxtrpl: new	48	new	113
\@newglossaryentryposthook: added		\@glsxtr@reg@glosslist: new	105
empty see value if not set and added		\makeglossaries: new	105
'see' to field key map	39	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new	46	\glxtrdiscardperiod: added check	
\cGlsformat: added	96	for first use	175
\cGlsformat: added	96	short-desc: fixed typo in	
\cGlsplformat: added	96	\glxtrinlinfullformat and	
\cGlsplformat: added	96	added missing second argument	211
\glstdisablehyper: added	80	1.02 (2016-04-25)	
\glstdohyperlink: added	79	\@glsxtr@current@style: new	50
\glstdonohyperlink: added	80	\Glsfmtfull: new	311

\glsfmtfull: new	311	\@GLSdescplural@: set abbreviation and regular format	65
\Glsfmtfullpl: new	312	\@GLSfirst@: set abbreviation format ..	62
\glsfmtfullpl: new	311	\@GLSfirstplural@: set abbreviation and regular format	63
\Glsfmtlong: new	310	\@GLSname@: set abbreviation and regular format	64
\glsfmtlong: new	310	\@GLSplural@: set abbreviation and regular format	63
\Glsfmtlongpl: new	310	\@GLSsymbol@: set regular format	65
\glsfmtlongpl: new	310	\@GLSsymbolplural@: set regular format	66
\Glsxtrheadfull: new	305	\@GLStext@: set abbreviation and regular format	61
\glsxtrheadfull: new	304	\@GLSuseri@: set regular format	66
\Glsxtrheadfullpl: new	305	\@GLSuserii@: set regular format	67
\glsxtrheadfullpl: new	304	\@GLSuseriii@: set regular format	67
\Glsxtrheadlong: new	303	\@GLSuseriv@: set regular format	67
\glsxtrheadlong: new	302	\@GLSuservi@: set regular format	68
\Glsxtrheadlongpl: new	303	\@GLSdesc@: set abbreviation and regular format	64
\glsxtrheadlongpl: new	303	\@GLSdescplural@: set abbreviation and regular format	65
\Glsxtrtitlefull: new	305	\@GLSfirst@: set abbreviation and regular format	62
\glsxtrtitlefull: new	304	\@GLSfirstplural@: set abbreviation and regular format	63
\Glsxtrtitlefullpl: new	305	\@GLSname@: set abbreviation and regular format	64
\glsxtrtitlefullpl: new	305	\@GLSplural@: set abbreviation and regular format	62
\Glsxtrtitlelong: new	303	\@GLSsymbol@: set regular format	65
\glsxtrtitlelong: new	302	\@GLSsymbolplural@: set regular format	66
\Glsxtrtitlelongpl: new	304	\@GLStext@: set abbreviation and regular format	61
\glsxtrtitlelongpl: new	303	\@GLSuseri@: set regular format	66
\ifglxtrinsertinside: new	200	\@GLSuserii@: set regular format	66
postfootnote: added redef of		\@GLSuseriii@: set regular format	67
\glsxtrsetupfulldefs	207	\@GLSuseriv@: set regular format	67
stylemods: new	21	\@GLSuservi@: set regular format	68
1.03 (2016-04-27)		\@GLSdesc@: set abbreviation and regular format	64
\@GLSfirstplural@: bug fix: misspelt cs name	63	\@GLSdescplural@: set abbreviation and regular format	62
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	63	\@GLSsymbol@: set regular format	65
\@GLSfirstplural@: bug fix: misspelt cs name	63	\@GLSsymbolplural@: set regular format	66
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	62	\@GLStext@: set abbreviation and regular format	61
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	62	\@GLSuseri@: set regular format	66
\glsxtrtitlelongpl: bug fix: changed \glsxtrlong to \glsxtrlongpl ..	303	\@GLSuserii@: set regular format	66
\glsxtrtitleshortpl: bug fix: changed \glsxtrshort to \glsxtrshortpl	297	\@GLSuseriii@: set regular format	67
1.04 (2015-04-30)		\@GLSuseriv@: set regular format	67
short-em-footnote: renamed from “footnote-em”	263	\@GLSuservi@: set regular format	68
1.04 (2016-05-02)		\@gls@preglossaryhook: added check for entry’s existence	173
\@glsxtrpostloctag: new	53	\@GLSdesc@: set abbreviation and regular format	64
\@GLSdesc@: set abbreviation and regular format	64	\@GLSdescplural@: set abbreviation and regular format	64
		\@GLSfirst@: set abbreviation and regular format	61

\@glsfirstplural@: set abbreviation and regular format	63	\glsxtruserparen: new	267
\@glsname@: set abbreviation and regular format	64	\glsxtrusersuffix: new	268
\@glsplural@: set abbreviation and regular format	62	\GlsXtrWarnDeprecatedAbbrStyle: new	199
\@glsymbol@: set regular format	65	short-em-long-em: new	251
\@glsymbolplural@: set regular format	65	short-em-long-em-desc: new	252
\@glstext@: set abbreviation and regular format	61	short-em-nolong: new	254
\@glsxtr@deprecated@abbrstyle: new	199	short-em-nolong-desc: new	255
\@glsxtr@do@style: new	21	short-em-postfootnote: renamed from “postfootnote-em”	265
\@glsxtr@doloctag: new	53	short-footnote: new	207
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand	114	short-long-user: new	274
\@glsxtr@pagetag: new	53	short-long-user-desc: new	275
\@glsxtr@pagetag: new	53	short-nolong: new	210
\@glsxtr@preloctag: new	53	short-nolong-desc: new	212
\@glsxtr@postloctag: new	53	short-postfootnote: new	209
\@glsxtr@preloctag: new	52, 53	short-sc-footnote: renamed from “footnote-sc”	227
\glossentrydesc: added glossdescfont attribute check	159	short-sc-nolong: new	222
\Glossentryname: added glossnamefont attribute check	163	short-sc-nolong-desc: new	223
\glossentryname: added glossnamefont attribute check	160	short-sc-postfootnote: renamed from “postfootnote-sc”	229
moved post name hook inside condition	162	short-sm-footnote: renamed from “footnote-sm”	242
\glsabbrvmfont: new	245	short-sm-nolong: new	236
\glsabbrvuserfont: new	267	short-sm-nolong-desc: new	238
\glsfirstabbrvmfont: new	245	short-sm-postfootnote: renamed from “postfootnote-sm”	243
\glsfirstabbrvuserfont: new	267	\letabbreviationstyle: new	199
\glsfirstlongemfont: new	245	\newabbreviationstyle: bug fix: corrected test for existence	198
\glsfirstlonguserfont: new	268	long-em-noshort-em: new	258
\glsifnotregularcategory: new ...	155	long-em-noshort-em-desc: new	261
\glslongdefaultfont: new	184	long-em-short-em: new	247
\glslongemfont: new	246	long-em-short-em-desc: new	248
\glslongfont: new	184	long-noshort: new	216
\glslonguserfont: new	267	long-noshort-desc: new	215
\glsxtrassignfieldfont: new	60	long-noshort-em: renamed from “long-em”	256
\GlsXtrEnablePreLocationTag: new .	52	long-noshort-em-desc: renamed from “long-desc-em”	260
\glsxtrfirstscfont: new	217	long-noshort-sc: renamed from “long-sc”	224
\glsxtrfirstsmfont: new	231	long-noshort-sc-desc: renamed from “long-desc-sc”	226
\glsxtrlongshortdescsort: new ...	202	long-noshort-sm: renamed from “long-sm”	238
\glsxtrpostnamehook: added category check	164	long-noshort-sm-desc: renamed from \long-desc-sm	240
\glsxtrregularfont: new	54		
\glsxtruserfield: new	267		

long-short-user: new	268	docdef option changed to choice	14
long-short-user-desc: new	274	\glxtr@usesee: new	39
\renewabbreviationstyle: new	198	\glxtrusesee: new	39
style: new	21	\glxtruseseeformat: new	39
1.05 (2016-06-10)		\if@glxtrdocdefrestricted: new ..	14
\eglssetwidest: new	364	1.07 (2016-08-15)	
\glFindWidestAnyName: new	367	\@@glxtrp: new	84
\glFindWidestAnyNameLocation:		\@GLSfirst@: added check for	
new	372	nohyperfirst attribute	62
\glFindWidestAnyNameSymbol: new	370	\@GLSfirstplural@: added check for	
\glFindWidestAnyNameSymbolLocation:		nohyperfirst attribute	63
new	371	\@GLSxtrp: new	85
\glFindWidestLevelTwo: new	368	\@Glsfirst@: added check for	
\glFindWidestUsedAnyName: new ..	367	nohyperfirst attribute	62
\glFindWidestUsedAnyNameLocation:		\@Glsfirstplural@: added check for	
new	372	nohyperfirst attribute	63
\glFindWidestUsedAnyNameSymbol:		\@Glsxtrp: new	85
new	369	\@gls@preglossaryhook: added	
\glFindWidestUsedAnyNameSymbolLocation:		\glossxtrsetpopts	174
new	370	\@glsfirst@: added check for	
\glFindWidestUsedLevelTwo: new .	367	nohyperfirst attribute	62
\glFindWidestUsedTopLevelName:		\@glsfirstplural@: added check for	
new	366	nohyperfirst attribute	63
\glfirstlongfootnotefont: new ..	205	\@glxtrinmark: new	294
\glsgetwidestname: new	366	\@glxtrnotinmark: new	294
\glsgetwidestsubname: new	366	\@glxtrp: new	84
\glslongfootnotefont: new	205	\@glxtrp@opt: new	84
\glxtrAltTreeIndent: new	364	\glossxtrsetpopts: new	84
\glxtralttreeInit: new	364	\glsp: new	87
\glxtrAltTreePar: new	364	\glsp: new	87
\glxtrAltTreeSetHangIndent: new	373	\glxtr@entry@p: new	85
\glxtrAltTreeSetSubHangIndent:		\glxtrabbrvfootnote: new	205
new	374	\glxtrchecknohyperfirst: new	61
\glxtralttreeSubSymbolDescLocation:		\glxtrfieldtitlecasecs: new	158
new	364	\glxtrifinmark: new	294
\glxtralttreeSymbolDescLocation:		\GLSxtrp: new	88
new	363	\Glsxtrp: new	87
\glxtrComputeTreeIndent: new ...	373	\glxtrp: new	86
\glxtrComputeTreeSubIndent: new	373	\glxtrsetpopts: new	84
\glxtrtreetopindent: new	364	short-long-desc: added text key	204
short-em-long: fixed incorrect font used		fixed misspelling of \glabbrvfont in	
by long form	250	plural key	204
\xglsetwidest: new	365	long-short-desc: added missing text	
1.06 (2016-06-18)		key	202
\@glsdoifexistsorwarn: new	14	fixed misspelling of \glabbrvfont .	202
\@glxtr@docdefval: new	14	footnote: changed first forms to use	
\@glxtr@usesee: new	39	\glfirstlongfootnotefont ...	205
General: disabled docdef key at the start		postfootnote: removed \footnote	
of the document	25	from first keys	207

switched from \glsfirstlongfont to	1.10 (2016-12-17)
\glsfirstlongfootnotefont ... 208	\@GLSp1@: fixed bug caused by typo in
\RestoreAcronyms: modified	command name 56
\@gls@link@checkfirsthyper to	1.11 (2017-01-19)
set \glsxtrifwasfirstuse 105	\@glsxtr@do@redef@for@gl@sentries:
1.08 (2016-12-13)	new 6
\@glsxtr@record: new 8	\@glsxtr@noidx@do: new 132
\@GLS@: added \@glsxtr@record 56	\@glsxtr@redef@for@gl@sentries: new . 6
\@GLSp1@: added \@glsxtr@record ... 56	\@glsxtr@shortcutsval: new 19
\@Gls@: added \@glsxtr@record 55	\@glsxtr@unsrt@getgroup@title: new 131
\@GLspl@: added \@glsxtr@record ... 55	\@print@noidx@glossary: added
\@gls@: added \@glsxtr@record 55	redefinition 115
\@gls@@link@: added	\glsxtr@addloc@listfield: added
\@glsxtr@record 56	group key 12
\@gls@field@link: added	added location key 12
\@glsxtr@record 54	\glsxtr@fields: new 123
\@gls@saveentrycounter: new 25	\glsxtr@linkprefix: new 124
\@glsdisp: added \@glsxtr@record .. 56	\glsxtr@org@new@ignored@glossary:
\@glspl@: added \@glsxtr@record ... 55	new 34
\@glsxtr@dorecord: new 10	\glsxtr@s@new@ignored@glossary: new 35
\@glsxtr@err@undefaction: new 6	\glsxtr@shortcutsval: new 124
\@glsxtr@record: new 7	\glsxtr@texencoding: new 124
\@glsxtr@warn@onexists@do: new ... 6	\glsxtr@writefields: new 124
\@glsxtr@warn@undefaction: new 6	\GlsXtrLoadResources: new 123
\@print@unsrt@glossary: new 128	\glsxtr@resourcefile: changed
General: added record package option ... 12	extension to .glstex 122
\glsadd: added \@glsxtr@record 60	\new@ignored@glossary: added starred
\glsdoifexists: now defines	version 34
\glslabel 37	1.12 (2017-02-03)
\glsxtr@do@wrglossary: new 25	\@glsxtr@recordcounter: new 10
\glsxtr@addloc@listfield: new 11	\@gls@preglossaryhook: check for
\glsxtr@indexonly@saveentrycounter:	definition 173
new 11	\@glsxtr@counter@recordhook: new . 126
\glsxtr@record: new 125	\@glsxtr@display@loc: new 116
\glsxtr@resource: new 123	\@glsxtr@docounter@record: new ... 126
\glsxtr@saveentrycounter: new 25	\@glsxtr@longnew@glossary@entry:
\glsxtr@setup@record: new 11	new 33
\glsxtr@assignfieldfont: added check	\@glsxtr@noop@recordcounter: new . 11
for existence 60	\@glsxtr@op@recordcounter: new ... 11
\glsxtr@resourcefile: new 122	\@glsxtr@provide@storagekey: new . 26
\printunsrt@glossaries: new 128	\@glsxtr@s@longnew@glossary@entry:
\printunsrt@glossary: new 127	new 33
1.09 (2016-12-16)	\@glsxtre@entryfmt: new 28
\@glsxtr@gettype: new 112	\@glsxtr@index@aliased: new 76
\@glsxtr@mixed@assign@sortkey:	\@glsxtr@set@alias@noindex: new 75
new 112	\@new@glossary@entry@posthook: added
\@print@glossary: redefined to save	check for alias key 43
options 111	\@no@glsxtr@index@aliased: new 76
\glsxtr@make@glossaries: new 112	\@print@unsrt@glossary: new 128

General: added target key to printgloss		\GlsXtrLoadResources: removed	
family	111	restriction on only one per document	123
\apptoglossarypreamble: new	32	\glxtrlocrangefmt: new	117
\csGlsXtrLetField: new	31	\glxtrpostlongdescription: new ..	34
\eGlsXtrSetField: new	32	\glxtrprovidestoragekey: new	26
\gGlsXtrSetField: new	31	\GlsXtrRecordCounter: new	126
\glsdohyperlink: added check for alias		\glxtrresourcecount: new	123
field	79	\glxtrresourcefile: added catcode	
\glsnoidxdisplayloc: added		change for @	123
redefinition	116	\glxtrsetaliasnoindex: new	75
\glissettoctitle: added patch	35	\GlsXtrSetField: new	31
\glxtr@counterrecord: new	125	\glxtrsetfieldifexists: new	31
\glxtr@langtag: new	124	\glxtrunsrtdo: new	131
\glxtr@newabbreviation: new	180	\Glsxtrusefield: new	31
\glxtr@org@newignoredglossary:		\glxtrusefield: new	30
Added check for existence	34	short-postlong-user: new	271
\glxtr@pluralsuffixes: new	124	short-postlong-user-desc: new ...	273
\glxtr@provideignoredglossary:		\longnewglossaryentry: added starred	
new	36	version	33
\glxtr@s@newignoredglossary:		long-postshort-user: new	269
Added check for existence	35	long-postshort-user-desc: new ...	270
\glxtr@s@provideignoredglossary:		postdot: new	15
new	36	\pretoglossarypreamble: new	32
\glxtrabbrvpluralsuffix: new ...	184	\print@noop@unsrtglossaryunit:	
\glxtralias: new	43	new	130
\glxtrcopytoglossary: new	37	\print@op@unsrtglossaryunit: new	130
\glxtrdeffield: new	31	\printunsrtglossary: added starred	
\glxtrdisplayendloc: new	117	form	127
\glxtrdisplayendlochook: new ...	117	\printunsrtglossaryhandler: new .	130
\glxtrdisplaysingleloc: new	116	\printunsrtglossaryunit: new	11
\glxtrdisplaystartloc: new	117	\printunsrtglossaryunitsetup: new	130
\glxtrdeffield: new	31	\provideignoredglossary: new	36
\glxtrentryfmt: new	28	\s@glxtr@provide@storagekey: new	26
\glxtrfielddolistloop: new	29	\s@printunsrtglossary: new	128
\glxtrfieldforlistloop: new	29	\xGlsXtrSetField: new	32
\glxtrfieldifinlist: new	29	1.13 (2017-02-07)	
\glxtrfieldlistadd: new	29	\@glsdisp: removed	
\glxtrfieldlistead: new	29	\@glxtr@org@glsdisp	56
\glxtrfieldlistgadd: new	29	\glxtrsetaliasnoindex: switched to	
\glxtrfieldlistxadd: new	29	\providecommand	75
\glxtrfieldxifinlist: new	29	1.14 (2017-04-18)	
\glxtrfmt: new	27	\@gls@link: added redefinition	58
\GlsXtrFmtDefaultOptions: new	27	\@gls@noidx@getgrouptitle: new ..	114
\GlsXtrFmtField: new	27	\@gls@removespaces: new	117
\glxtrifkeydefined: new	26	\@glxtr@do@automake@err: new ...	125
\glxtrindexaliased: new	76	\@glxtr@org@gloautosee: new	23
\GlsXtrLetField: new	31	\@glxtr@record: added third arg	7
\GlsXtrLetFieldToField: new	31	\@glxtr@recordsee: new	11

General: added \glsadd option		1.16 (2017-06-15)	
theHvalue	60	\@glo@autosee: added redefinition	24
added \glsadd option thevalue	60	\@gls@noidx@getgrouptitle: fixed	
\glsdisablehyper: added redefinition .	80	bug	114
\glsenableentrycount: fixed		\@glsxtr@addunusedxrefs: added	
assignment of \@cGls@	92	check for seealso field	44
\glsenableentryunitcount: fixed		\@glsxtr@checkgroup: use \csuse	
assignment of \@cGls@	100	instead of \csname	131
\glsnavigation: new	115	\@glsxtr@dorecordnodefer: new	10
\glsxtr@org@getgrouptitle: new ..	114	\@print@unsrt@glossary: corrected	
\glsxtr@recordsee: new	7	misspelt command	128
\glsxtr@writefields: added check for		\@printunsrt@glossary@handler:	
automake	125	new	130
\glsxtrdisplayendloc: added check		General: added check for	
for empty format	117	\@gls@setupsort@none	13
\glsxtrgetgrouptitle: new	114	\@gls@checkseeallowed: added	
\glsxtrinitwrgloss: new	57	redefinition	24
\glsxtrlocationhyperlink: new ...	117	\@glsxtr@writefields: added	
\glsxtrsetgrouptitle: new	115	\providecommand lines	124
\glsxtrsupphypernumber: new	118	\@glsxtrautoindex: new	167
\ifglsxtrwrglossbefore: new	57	\@glsxtrautoindexentry: new	167
1.15 (2017-05-10)		\@glsxtrautoindexsort: new	168
\@glsxtr@dorecord: corrected		\@glsxtrindexseealso: new	40
premature expansion of \@glslocref	10	\@glsxtrseealsolabels: new	43
short-em-long-em: fixed spelling of		\@glsxtrseelist: new	40
\glsabbrvfont	251	\@glsxtruseseealso: new	40
short-long: fixed spelling of		\@glsxtruseseealsoformat: new	40
\glsabbrvfont	203	\@seealso: new	40
short-long-user: fixed spelling of		autoseeindex: new	15
\glsabbrvfont	274	1.17 (2017-08-09)	
short-postlong-user: fixed spelling of		\@glsxtr@mark@wordseps: new	179
\glsabbrvfont	271	\@glsxtr@markwordseps: new	179
short-postlong-user-desc: fixed		\@glsxtr@noidx@displaynumberlist:	
spelling of \glsabbrvfont	273	replace hard-coded ?? with	
long-em-short-em: fixed spelling of		\glsxtrundeftag	113
\glsabbrvfont	248	\@glsxtr@noidx@entrynumberlist:	
long-postshort-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	269	\glsxtrundeftag	113
long-postshort-user-desc: fixed		\@glsxtr@noidx@numberlistloop:	
spelling of \glsabbrvfont	271	replace hard-coded ?? with	
long-short: fixed spelling of		\glsxtrundeftag	113
\glsabbrvfont	201	\@glsxtrifhyphenstart: new	276
long-short-user: fixed spelling of		General: removed some inconsistencies	
\glsabbrvfont	268	in the abbreviation styles	200
footnote: fixed spelling of		\glsabbrvhyphenfont: new	277
\glsabbrvfont	205	\glsabbrvonlyfont: new	290
postfootnote: fixed spelling of		\glsabbrvscfont: new	217
\glsabbrvfont	207	\glsabbrvsmfont: new	231

\glsabbrvuserfont: initialised to default font	267	short-long-user-desc: corrected first forms	275
\glsfirstabbrvhyphenfont: new ...	277	short-nolong-desc-noreg: new ...	212
\glsfirstabbrvonlyfont: new	290	short-nolong-noreg: new	210
\glsfirstabbrvscfont: new	217	long-em-noshort-em-desc-noreg: new	263
\glsfirstabbrvsmfont: new	231	long-em-noshort-em-noreg: new ...	259
\glsfirstlonghyphenfont: new ...	277	long-hyphen-noshort-desc-noreg: new	279
\glsfirstlongonlyfont: new	290	long-hyphen-postshort-hyphen: new	282
\glslonghyphenfont: new	277	long-hyphen-postshort-hyphen-desc: new	284
\glslongonlyfont: new	290	long-hyphen-short-hyphen: new ...	277
\glslonguserfont: initialised to default font	267	long-hyphen-short-hyphen-desc: new	278
\glsxtr@newabbreviation: added \glsxtrorgshort and \glsxtrorglong	180	long-noshort-desc-noreg: new ...	215
\GlsXtrDefineAcShortcuts: new	18	long-noshort-noreg: new	216
\glsxtrgenabbrvfmt: added check for \ifglsxtrininsertinside	194	long-only-short-only: new	291
\glsxtrhyphensuffix: new	277	long-only-short-only-desc: new ..	292
\glsxtrifhyphenstart: new	276	long-short-user-desc: corrected first forms	274
\glsxtrlonghyphen: new	281	1.18 (2017-08-10)	
\glsxtrlonghyphennoshort: new ...	279	stylemods: changed default value to “default”	21
\glsxtrlonghyphenshort: new	276	1.19 (2017-09-09)	
\glsxtrlongshortdescname: new ...	202	\@glsxtr@defaultnumberformat: new .	7
\glsxtronlydescname: new	292	\@glsxtr@dorecord: Use \@glsrecordlocref instead of \@glslocref	10
\glsxtronlydescsort: new	292	\@glsxtr@dorecordnodefer: Use \theglscopycounter for the location rather than \@glslocref ..	10
\glsxtronlysuffix: new	290	\@glsxtr@record@setting: new	12
\glsxtrparen: new	182	\@glsxtr@record@setting@alsoindex: new	12
\glsxtrposthyphenlong: new	287	\@glsxtrifhasfield: new	30
\glsxtrposthyphenshort: new	282	General: added \glslink option theHvalue	57
\glsxtrposthyphensubsequent: new	282	added \glslink option thevalue ...	57
\glsxtrshortdescname: new	211	\glsxtr@writefields: removed double-quotes around \jobname ..	125
\glsxtrshorthyphen: new	287	\glsxtrdoautoindexname: changed format test	167
\glsxtrshorthyphenlong: new	285	\glsxtrhyperlink: new	79
\glsxtrshortlongdescname: new ...	204	\glsxtrifhasfield: new	30
\glsxtrshortlongdescsort: new ...	204	\GlsXtrSetDefaultNumberFormat: new	7
\Glsxtrsubsequentfmt: new	197	\s@glsxtrifhasfield: new	30
\glsxtrsubsequentfmt: new	196		
\Glsxtrsubsequentplfmt: new	197		
\glsxtrsubsequentplfmt: new	196		
\glsxtrword: new	179		
\glsxtrwordsep: new	179		
short-hyphen-long-hyphen: new ...	285		
short-hyphen-long-hyphen-desc: new	286		
short-hyphen-postlong-hyphen: new	287		
short-hyphen-postlong-hyphen-desc: new	289		

headings	378	\glxtrbookindexbetween: new	383
redefined		\glxtrbookindexbookmark: new ...	383
mcoltreenonamehypergroup to		\glxtrbookindexcols: new	382
discourage breaks after group		\glxtrbookindexcolspread: new ..	383
headings	378	\glxtrbookindexfirstmark: new ..	387
redefined mcoltreenonamespannav to		\glxtrbookindexfirstmarkfmt: new	387
discourage breaks after group		\glxtrbookindexformatheader: new	383
headings	379	\glxtrbookindexgroupskip: new ..	383
redefined mcoltreepspannav to		\glxtrbookindexlastmark: new ...	387
discourage breaks after group		\glxtrbookindexlastmarkfmt: new	387
headings	377	\glxtrbookindexmarkentry: new ..	387
redefined treegroup to discourage		\glxtrbookindexname: new	382
breaks after group headings	361	\glxtrbookindexparentchildsep:	
redefined treehypergroup to		new	382
discourage breaks after group		\glxtrbookindexparentschildsep:	
headings	362	new	382
redefined treenonamegroup to		\glxtrbookindexprelocation: new	382
discourage breaks after group		\glxtrbookindexsubatendgroup:	
headings	363	new	383
redefined treenonamehypergroup to		\glxtrbookindexsubbetween: new ..	383
discourage breaks after group		\glxtrbookindexsubname: new	382
headings	363	\glxtrbookindexsubprelocation:	
debug: new	22	new	382
\gglsssetwidest: new	364	\glxtrbookindexsubsubatendgroup:	
\glsdisablehyper: added check for		new	383
existence	80	\glxtrbookindexsubsubbetween:	
changed to use \def rather than \let .	80	new	383
\glsenablehyper: changed to use \def		\glxtrbookindexthepage: new	386
rather than \let	80	\glxtrdetoklocation: new	135
\Glsfmtname: new	307	\glxtrenablerecordcount: new ...	135
\glsfmtname: new	307	\glxtrglossentry: new	126
\glshex: new	313	\glxtrgroupfield: new	131
\glslstchildpostlocation: new ..	349	\Glsxtrheadname: new	298
\glslstchildprelocation: new ...	349	\glxtrheadname: new	298
\glslstprelocation: new	349	\GlsXtrIfFieldEqStr: new	32
\glsnahyperlink: patched	78	\glxtriflabelinlist: new	130
\glssseeitemformat: new	40	\glxtrifrecordtrigger: new	136
\glsshows target: new	23	\glxtrindexseealso: added check	
\glstreechildprelocation: new ...	359	that the entry exists	41
\glstreeprelocation: new	359	\glxtrtrinityhyperoutside: new	57
\glstriggerrecordformat: new	137	\GlsXtrLocationRecordCount: new ..	135
\glseuseabbrvfont: new	194	\glxtrnewgls: new	133, 134
\glseuselongfont: new	194	\glxtrnewGLSlike: new	134
\glxtr@do@alsoindex@wrglossary:		\glxtrnewglslike: new	134
new	8	\glxtrnewrgls: new	134
\glxtr@org@@do@wrglossary: new ..	25	\glxtrnewrGLSlike: new	135
\glxtr@org@dohyperlink: new	78	\glxtrnewrglslike: new	134
\glxtr@setbookindexmark: new ...	387	\glxtrprelocation: new	348, 382
\glxtrbookindexatendgroup: new ..	383	\GlsXtrRecordCount: new	135

\glxtrrecordtriggervalue: new ..	135	\glseeitemformat: switched check	
\glxtrresourcefile: now disables		from regular to short	40
record key	123	\glxtr@setaccessdisplay: new ...	164
\glxtrresourceinit: new	123	\glxtr@writefields: provide	
\GlsXtrSetRecordCountAttribute:		\glxtr@record in aux file	124
new	136	\glxtractivatenopost: new	110
\glxtrtitlename: new	298	\glxtrbookindexprelocation:	
\glxtrtitleorpdforheading: new .	294	removed check for no post dot	382
\GlsXtrTotalRecordCount: new	135	\glxtrglossentryother: new	127
\glxtrwrglossmark: new	22	\glxtrnopostpunc: new	110
short-em: new	253	1.23 (2017-11-12)	
short-sc: corrected first letter		\@@glxtrfmt: added check for indexing	28
uppercasing	221	added grouping	27
short-sm: corrected first letter		new	27
uppercasing	236	\@glxtr@nopostpunc@postdesc: new	111
\ifglxtr@hyperoutside: new	57	\@glxtr@restore@postpunc: new ..	111
all: new	347	\@glxtrentryfmt: fixed missing label	
nolong-short: new	212	argument	28
nolong-short-em: new	255	\@glxtrfmt: new	27
nolong-short-noreg: new	213	\eglupdatewidest: new	365
nolong-short-sc: new	223	\gglupdatewidest: new	365
nolong-short-sm: new	238	\glupdatewidest: new	365
nopostdot: new	15	\GlsXtrDefineAbbreviationShortcuts:	
postpunc: new	16	changed \newabbr definition to use	
\printunsrtglossaryentryprocesshook:		\providecommand	17
new	129	\GlsXtrDefineAcShortcuts: changed	
\printunsrtglossarypredoglossary:		\newabbr definition to use	
new	129	\providecommand	18
\rGLS: new	139	\glxtrfmtdisplay: new	28
\rGls: new	138	\glxtrifcustomdiscardperiod: new	175
\rgls: new	137	\GlsXtrIfFieldUndef: new	30
\rGLSformat: new	140	\glxtrrestorepostpunc: new	111
\rGlsformat: new	140	\s@glxtrfmt: new	27
\rglsformat: new	139	\s@glxtrfmt: new	27
\rGLSpl: new	139	\xglupdatewidest: new	365
\rGlspl: new	138	1.24 (2017-11-14)	
\rglspl: new	137	\gl sadd: added \@gls@setsort	60
\rGLSplformat: new	140	\glxtrforcsvfield: new	30
\rGlsplformat: new	140	\glxtrlocalsetgrouptitle: new ..	115
\rglsplformat: new	140	1.25 (2017-11-14)	
\s@glxtrifhasfield: switched from		\glxtrbookindexmulticolseenv: new	383
\ifdef to \ifundef	30	1.25 (2017-11-24)	
1.22 (2017-11-08)		\glsextrapostnamehook: new	164
\@glxtr@nopostpunc: new	110	\glxtrfootnotename: new	205
\@glxtr@orgprintglossary: changed		\glxtrlongnoshortdescname: new .	214
explicit \let for \nopostdesc to		\glxtrlongnoshortname: new	216
\glxtractivatenopost	110	\glxtrlongshortname: new	200
\@glxtrglossentryother: new	127	\glxtrlongshortuserdescname: new	270
\glossentrynameother: new	165	\glxtronlyname: new	290

\glxtrpostlinkAddDescOnFirstUse: changed to use \glxtrparen	175	\glxtrGeneralLatinVIrules: new ..	325
\glxtrpostlinkAddSymbolOnFirstUse: changed to use \glxtrparen	175	\glxtrGeneralLatinVrules: new ..	324
\glxtrshortlongname: new	202	\glxtrgeneralpuncIIrules: new ..	321
\glxtrshortlonguserdescname: new	273	\glxtrgeneralpuncIrules: new ...	320
\glxtrshortnolongname: new	209	\glxtrgeneralpuncrules: new	320
1.26 (2018-01-05)		\glxtrhyphenrules: new	319
\@glxtr@do@inc@linkcount: new ..	140	\glxtrLatinA: new	327
\glslinkpresetkeys: new	58	\glxtrLatinAA: new	329
\glxtr@inc@linkcount: new	57	\glxtrLatinAELigature: new	329
\GlsXtrEnableLinkCounting: new ..	142	\glxtrLatinE: new	327
\GlsXtrIfLinkCounterDef: new	141	\glxtrLatinEszettSs: new	328
\glxtrinclinkcounter: new	141	\glxtrLatinEszettSz: new	329
\GlsXtrLinkCounterName: new	141	\glxtrLatinEth: new	329
\GlsXtrLinkCounterValue: new	141	\glxtrLatinH: new	327
\GlsXtrTheLinkCounter: new	141	\glxtrLatinI: new	327
1.27 (2018-02-26)		\glxtrLatinInsularG: new	329
\@glxtrdialecthook: new	25	\glxtrLatinK: new	327
General: added		\glxtrLatinL: new	328
glossaries-extra-bib2gls.sty	313	\glxtrLatinLslash: new	330
\Alpha: new	313	\glxtrLatinM: new	328
\Beta: new	314	\glxtrLatinN: new	328
\Chi: new	314	\glxtrLatinO: new	328
\Digamma: new	314	\glxtrLatinOELigature: new	329
\Epsilon: new	314	\glxtrLatinOslash: new	329
\Eta: new	314	\glxtrLatinP: new	328
\glxtr@loaddialect: new	312	\glxtrLatinS: new	328
\glxtrBasicDigitrules: new	344	\glxtrLatinSchwa: new	328
\glxtrcombiningdiacriticIIrules: new	318	\glxtrLatinT: new	328
\glxtrcombiningdiacriticIrules: new	318	\glxtrLatinThorn: new	329
\glxtrcombiningdiacriticIrules: new	317	\glxtrLatinWynn: new	329
\glxtrcombiningdiacriticIVrules: new	319	\glxtrLatinX: new	328
\glxtrcombiningdiacriticrules: new	317	\glxtrMathGreekIIrules: new	336
\glxtrcontrolrules: new	316	\glxtrMathGreekIrules: new	335
\glxtrcurrencyrules: new	321	\glxtrMathItalicAlpha: new	340
\glxtrdigitrules: new	343	\glxtrMathItalicBeta: new	340
\glxtrfractionrules: new	344	\glxtrMathItalicChi: new	343
\glxtrGeneralLatinIIrules: new	323	\glxtrMathItalicDelta: new	340
\glxtrGeneralLatinIrules: new ..	322	\glxtrMathItalicEpsilon: new ...	340
\glxtrGeneralLatinIrules: new ..	321	\glxtrMathItalicEta: new	341
\glxtrGeneralLatinIVrules: new ..	323	\glxtrMathItalicGamma: new	340
\glxtrGeneralLatinVIIrules: new	326	\glxtrMathItalicGreekIIrules: new	331
\glxtrGeneralLatinVIIrules: new	326	\glxtrMathItalicGreekIrules: new	331
		\glxtrMathItalicIota: new	341
		\glxtrMathItalicKappa: new	341
		\glxtrMathItalicLambda: new	341
		\glxtrMathItalicLowerGreekIIrules: new	334

<code>\glstrMathItalicLowerGreekIrules:</code>	<code>\glstrUpLambda: new</code>	338
new	<code>\glstrUpMu: new</code>	338
<code>\glstrMathItalicMu: new</code>	<code>\glstrUpNu: new</code>	338
<code>\glstrMathItalicNabla: new</code>	<code>\glstrUpOmega: new</code>	340
<code>\glstrMathItalicNu: new</code>	<code>\glstrUpOmicron: new</code>	339
<code>\glstrMathItalicOmega: new</code>	<code>\glstrUpPhi: new</code>	339
<code>\glstrMathItalicOmicron: new</code> ...	<code>\glstrUpPi: new</code>	339
<code>\glstrMathItalicPartial: new</code> ...	<code>\glstrUpPsi: new</code>	340
<code>\glstrMathItalicPhi: new</code>	<code>\glstrUpRho: new</code>	339
<code>\glstrMathItalicPi: new</code>	<code>\glstrUpSigma: new</code>	339
<code>\glstrMathItalicPsi: new</code>	<code>\glstrUpTau: new</code>	339
<code>\glstrMathItalicRho: new</code>	<code>\glstrUpTheta: new</code>	338
<code>\glstrMathItalicSigma: new</code>	<code>\glstrUpUpsilon: new</code>	339
<code>\glstrMathItalicTau: new</code>	<code>\glstrUpXi: new</code>	339
<code>\glstrMathItalicTheta: new</code>	<code>\glstrUpZeta: new</code>	338
<code>\glstrMathItalicUpperGreekIIrules:</code>	<code>\Iota: new</code>	314
new	<code>\Kappa: new</code>	314
<code>\glstrMathItalicUpperGreekIrules:</code>	<code>\Mu: new</code>	314
new	<code>\Nu: new</code>	314
<code>\glstrMathItalicUpsilon: new</code> ...	<code>\Omicron: new</code>	314
<code>\glstrMathItalicXi: new</code>	<code>\omicron: new</code>	314
<code>\glstrMathItalicZeta: new</code>	<code>\Rho: new</code>	314
<code>\glstrMathUpGreekIIrules: new</code> ..	<code>\Tau: new</code>	314
<code>\glstrMathUpGreekIrules: new</code> ...	<code>\Upalpha: new</code>	315
<code>\glstrnonprintablerules: new</code> ...	<code>\Upbeta: new</code>	315
<code>\glstrprovidecommand: new</code>	<code>\Upchi: new</code>	315
<code>\glstrspacerules: new</code>	<code>\Upepsilon: new</code>	315
<code>\glstrSubScriptDigitrules: new</code> .	<code>\Upeta: new</code>	315
<code>\glstrSuperScriptDigitrules: new</code>	<code>\Upiota: new</code>	315
<code>\glstrUpAlpha: new</code>	<code>\Upkappa: new</code>	315
<code>\glstrUpBeta: new</code>	<code>\Upmu: new</code>	315
<code>\glstrUpChi: new</code>	<code>\Upnu: new</code>	315
<code>\glstrUpDelta: new</code>	<code>\Upomicron: new</code>	315
<code>\glstrUpDigamma: new</code>	<code>\upomicron: new</code>	315
<code>\glstrUpEpsilon: new</code>	<code>\Uprho: new</code>	315
<code>\glstrUpEta: new</code>	<code>\Uptau: new</code>	315
<code>\glstrUpGamma: new</code>	<code>\Upzeta: new</code>	315
<code>\glstrUpIota: new</code>	<code>\Zeta: new</code>	314
<code>\glstrUpKappa: new</code>		

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	
\.	15, 16, 175, 359
\@	123
\@cGLS@	92, 100
\@cGLSpl@	92, 100
\@cGls@	92, 100
\@cGlspl@	92, 100
\@cgls@	92, 100
\@cglspl@	92, 100
\@do@wrglossary	8, 9, 107
\@do@wrglossary	11, 13, 25, 60, 76
\@glo@assign@sortkey	112
\@glo@list	6
\@glo@type	128
\@glossarysec	383
\@gls@expand@field	27
\@glslocalreset	90
\@glslocalunset	89
\@glsreset	90
\@glsunset	89
\@glsxtr@autoindex@escspch	169, 170
\@glsxtr@checkspch	168–171
\@glsxtr@disabledflycommand	49
\@glsxtr@org@postdescription	110, 111
\@glsxtr@record	13
\@glsxtr@recordcounter	13, 126
\@glsxtrfmt	27
\@glsxtrp	84, 85
\@glsxtrpostloctag	52
\@glsxtrpreloctag	52
\@glsxtrwrglossmark	8, 11, 23, 25, 41, 45, 106
\@newglossaryentry@defcounters	91
\@newglossaryentry@defunitcounters	99
\@par	364
\@ACRlong	81
\@ACRlongpl	81
\@ACRshort	81
\@ACRshortpl	81
\@Acrlong	81
\@Acrlongpl	81
\@Acrshort	81
\@Acrshortpl	81
\@GLS@	81, 94, 95, 139
\@GLSdesc@	65
\@GLSpl@	81, 95, 96, 139
\@GLSplural@	82
\@GLSsymbol@	66
\@GLStext@	81
\@GLSxtr@full	186
\@GLSxtr@fullpl	187
\@GLSxtr@p@acrlong@	81
\@GLSxtr@p@acrlongpl@	81
\@GLSxtr@p@acrshort@	81
\@GLSxtr@p@acrshortpl@	81
\@GLSxtr@p@long@	81
\@GLSxtr@p@longpl@	81
\@GLSxtr@p@plural@	81
\@GLSxtr@p@short@	81
\@GLSxtr@p@shortpl@	81
\@GLSxtr@p@text@	81
\@GLSxtrlong	81, 190
\@GLSxtrlongpl	81, 193, 194
\@GLSxtrp	88, 89
\@GLSxtrshort	81, 189
\@GLSxtrshortpl	81, 192
\@Gls@	81, 94, 95, 138
\@Gls@acentryname	103
\@Gls@entry@field	72, 87, 88, 165
\@Gls@entryname	103
\@GlsXtrEnableOnTheFly	46, 47
\@Glspl@	81, 94, 95, 139
\@Glsplural@	82
\@Glstext@	81
\@Glsxtr	48, 49

\@Glsxtr@full	185	\@endfortrue	30, 164, 197
\@Glsxtr@fullpl	187	\@firstofone	61, 129, 159, 160, 166, 172
\@Glsxtr@p@acrlong@	81	\@firstofthree	56, 60, 68–71, 77, 78, 185, 186, 188, 189, 191, 193
\@Glsxtr@p@acrlongpl@	81	\@firstoftwo	. 62, 63, 65, 66, 69–72, 74, 77, 105, 164, 165, 176, 177, 185–187, 191–194, 294, 295
\@Glsxtr@p@acrshort@	81	\@for	6, 21, 30, 44, 91, 102, 106, 109, 115, 128, 136, 142, 158, 164, 172
\@Glsxtr@p@acrshortpl@	81	\@glo@alias	42, 43
\@Glsxtr@p@long@	81	\@glo@assign@sortkey	108
\@Glsxtr@p@longpl@	81	\@glo@autosee	23
\@Glsxtr@p@plural@	81	\@glo@autoseehook	42
\@Glsxtr@p@short@	81	\@glo@category	96
\@Glsxtr@p@shortpl@	81	\@glo@check@sortallowed	109
\@Glsxtr@p@text@	81	\@glo@counterprefix	10, 117
\@Glsxtrlong	81, 190	\@glo@countunit	96, 97
\@Glsxtrlongpl	81, 193	\@glo@default@sorttype	108
\@Glsxtrp	87, 88	\@glo@desc	33, 34
\@Glsxtrpl	49	\@glo@descplural	33, 34
\@Glsxtrshort	81, 188	\@glo@group	12
\@Glsxtrshortpl	81, 191	\@glo@label 11, 12, 26, 39, 42–44, 72, 80, 366–373
\@acrlong	81	\@glo@location	12
\@acrlongpl	81	\@glo@loclist	11
\@acrshort	81	\@glo@name	167
\@acrshortpl	81	\@glo@no@assign@sortkey	112
\@addtoreset	141	\@glo@parent	368, 369
\@afterheading 350, 351, 360–363, 376–379, 386	\@glo@see	39, 40, 43, 44
\@alt@glshyp@opt	77	\@glo@seealso	42
\@auxout	10, 11, 45, 53, 92, 101, 106, 118, 119, 123–126, 387	\@glo@sort	167
\@bibgl@restoreat	123	\@glo@sorttype	108, 115
\@cGLS	95	\@glo@text	56
\@cGLS@	92, 95, 100	\@glo@thislettergrp	131, 132
\@cGLSpl	95	\@glo@thisvalue	267
\@cGLSpl@	92, 95, 100	\@glo@tmp	26, 40, 72
\@cGls@	92, 100	\@glo@type	44, 78, 103, 106, 109, 110, 112, 115, 116, 118, 119, 121, 122, 128, 129
\@cGlspl@	92, 100	\@glo@types	156, 366–372
\@cgls@	92, 100	\@glossary@default@style	50, 109, 381
\@cglspl@	92, 100	\@glossarystyle	109, 110
\@disable@onlypremakeg	106	\@gls@	81, 93, 95, 137
\@do@auxoutstuff	118, 119	\@gls@@link	56
\@do@gl@getcounterprefix	10	\@gls@ReturnAfterFi	117
\@do@gl@see	42, 43	\@gls@actualchar	168
\@do@newglossaryentry	103, 182	\@gls@adjustmode	60
\@do@seeglossary	13, 23, 45, 106	\@gls@alt@hyp@opt	78
\@do@wrglossary	59, 137	\@gls@alt@hyp@opt@char	77, 78
\@empty	60, 68–72, 111, 168, 169, 185–194	\@gls@alt@hyp@opt@keys	78
\@end@glxtr@addunused	44		
\@end@glxtr@gettype	108, 112		
\@end@glxtr@usesee	39		
\@end@glxtr@trifhyphenstart	276		

<code>\@gls@automake</code>	109	<code>\@gls@noidxloclist@finalsep</code>	112
<code>\@gls@between</code>	115	<code>\@gls@noidxloclist@prev</code>	112
<code>\@gls@checkedmkidx</code>	168–171	<code>\@gls@noidxloclist@sep</code>	112
<code>\@gls@checkmkidxchars</code>	41, 167	<code>\@gls@noref@warn</code>	107, 116
<code>\@gls@codepage</code>	119	<code>\@gls@org@glsnoidxdisplayloc</code>	113
<code>\@gls@counter</code>	9, 10, 58, 60, 76, 136	<code>\@gls@org@glsseeformat</code>	113
<code>\@gls@currentlettergroup</code>	116, 128, 131, 132	<code>\@gls@preglossaryhook</code>	110, 172
<code>\@gls@declareoption</code>	5	<code>\@gls@prevlevel</code>	374, 375, 379, 380
<code>\@gls@default@longpl</code>	180, 181	<code>\@gls@quotechar</code>	168
<code>\@gls@doautomake</code>	109, 125	<code>\@gls@reference</code>	45, 106, 107
<code>\@gls@doautomake@err</code>	125	<code>\@gls@saveentrycounter</code>	13, 25, 58, 60, 136
<code>\@gls@encapchar</code>	168	<code>\@gls@see@noindex</code>	24, 123
<code>\@gls@entry@count</code>	92	<code>\@gls@setdefault@glslink@opts</code>	9, 28, 58, 76
<code>\@gls@entry@field</code>	26, 30, 31, 42, 72, 86–89, 91, 127	<code>\@gls@setsort</code>	59, 60
<code>\@gls@entry@unitcount</code>	100, 101	<code>\@gls@setupsort@none</code>	13
<code>\@gls@field@font</code>	61–68	<code>\@gls@short</code>	181
<code>\@gls@field@link</code>	61–68, 73, 74	<code>\@gls@shortpl</code>	178, 181, 182
<code>\@gls@getcounterprefix</code>	10	<code>\@gls@sort</code>	131
<code>\@gls@getgrouptitle</code>	114, 115, 128	<code>\@gls@thisval</code>	164, 165
<code>\@gls@grptitle</code>	78, 115	<code>\@gls@tmp</code>	115
<code>\@gls@hyp@opt</code>	73, 74, 78, 95, 133, 137–139, 184–193	<code>\@gls@tmpb</code>	170, 171
<code>\@gls@hyp@opt@cs</code>	77, 78	<code>\@gls@type</code>	106, 107, 109, 197, 366–373
<code>\@gls@ifinlist</code>	130	<code>\@gls@write@entrycounts</code>	92
<code>\@gls@increment@currcount</code>	91	<code>\@gls@write@entryunitcounts</code>	100
<code>\@gls@increment@currunitcount</code>	99	<code>\@gls@write@entryunitcounts@do</code>	101
<code>\@gls@keymap</code> ..	11, 12, 26, 39, 42, 72, 124, 164	<code>\@gls@xref</code>	11, 41
<code>\@gls@label</code> ..	8–10, 45, 77, 106, 107, 126, 197	<code>\@gls@abbrv@current@abbreviation</code>	180, 194
<code>\@gls@levelchar</code>	168	<code>\@gls@acronymlists</code>	103
<code>\@gls@link</code>	28, 55, 56, 68–72, 185–194	<code>\@gls@doifexistsorwarn</code>	14, 160, 161, 163, 165
<code>\@gls@link@checkfirsthyper</code>	56, 105	<code>\@gls@entry</code>	92, 101
<code>\@gls@link@label</code>	58, 136	<code>\@gls@link</code>	59, 78, 80
<code>\@gls@link@nocheckfirsthyper</code>	55, 68–72, 185–194	<code>\@gls@nextpages</code>	110
<code>\@gls@link@opts</code>	58	<code>\@gls@nonextpages</code>	110
<code>\@gls@list</code>	115	<code>\@gls@numberformat</code>	9, 10, 58, 60, 76, 136, 164, 167
<code>\@gls@local@increment@currcount</code>	91	<code>\@gls@order</code>	106
<code>\@gls@local@increment@currunitcount</code>	100	<code>\@gls@pl@</code>	81, 94, 95, 138
<code>\@gls@location</code>	132, 133	<code>\@gls@plural@</code>	81
<code>\@gls@loclist</code>	112, 113, 132	<code>\@gls@punc@token</code>	177
<code>\@gls@long</code>	180	<code>\@gls@recordlocref</code>	10
<code>\@gls@longpl</code>	178, 180–182	<code>\@gls@showtarget</code>	79
<code>\@gls@map</code>	164	<code>\@gls@style@altlist</code>	349
<code>\@gls@nohyperlist</code>	35, 36	<code>\@gls@style@altlistgroup</code>	350
<code>\@gls@noidx@do</code>	116	<code>\@gls@style@altlisthypergroup</code>	351
<code>\@gls@noidx@getgrouptitle</code>	128	<code>\@gls@style@alttree</code>	363
<code>\@gls@noidx@nosanitizesort</code>	108	<code>\@gls@style@alttreegroup</code>	375
<code>\@gls@noidx@sanitizesort</code>	108	<code>\@gls@style@alttreehypergroup</code>	375
		<code>\@gls@style@index</code>	359

<code>\@glsstyle@indexgroup</code>	360	<code>\@glsxtr@autoindex@setname</code>	167
<code>\@glsstyle@indexhypergroup</code>	360	<code>\@glsxtr@autoindexcrossrefs</code>	13, 14, 39, 42
<code>\@glsstyle@inline</code>	359	<code>\@glsxtr@autoseeindexfalse</code>	13
<code>\@glsstyle@list</code>	349	<code>\@glsxtr@autoseeindextrue</code>	15
<code>\@glsstyle@listdotted</code>	348	<code>\@glsxtr@bookindex@atendgroup</code> ..	384–386
<code>\@glsstyle@listgroup</code>	350	<code>\@glsxtr@bookindex@atsubendgroup</code> ..	385
<code>\@glsstyle@listhypergroup</code>	350	<code>\@glsxtr@bookindex@atsubsubendgroup</code>	385
<code>\@glsstyle@mcolalttree</code>	379	<code>\@glsxtr@bookindex@between</code>	384, 386
<code>\@glsstyle@mcolalttreegroup</code>	379	<code>\@glsxtr@bookindex@sep</code>	384, 385
<code>\@glsstyle@mcolalttreehypergroup</code> ..	380	<code>\@glsxtr@bookindex@subatendgroup</code> ..	
<code>\@glsstyle@mcolalttreespannav</code>	380	384–386
<code>\@glsstyle@mcolindexgroup</code>	376	<code>\@glsxtr@bookindex@subbetween</code> ..	384, 385
<code>\@glsstyle@mcolindexhypergroup</code>	376	<code>\@glsxtr@bookindex@subsep</code>	384, 385
<code>\@glsstyle@mcolindexspannav</code>	376	<code>\@glsxtr@bookindex@subsubatendgroup</code>	
<code>\@glsstyle@mcoltreegroup</code>	377	384–386
<code>\@glsstyle@mcoltreehypergroup</code>	377	<code>\@glsxtr@bookindex@subsubbetween</code> ..	
<code>\@glsstyle@mcoltreenamegroup</code>	378	384, 385
<code>\@glsstyle@mcoltreenamehypergroup</code>	378	<code>\@glsxtr@bookindex@groupskip</code> ...	384, 386
<code>\@glsstyle@mcoltreenamespannav</code> ..	379	<code>\@glsxtr@cat</code>	91, 102, 136, 172
<code>\@glsstyle@mcoltreespannav</code>	377	<code>\@glsxtr@checkgroup</code>	129
<code>\@glsstyle@tree</code>	361	<code>\@glsxtr@counterrecordhook</code>	10, 11
<code>\@glsstyle@treegroup</code>	361	<code>\@glsxtr@csname</code>	97, 98, 100
<code>\@glsstyle@treehypergroup</code>	362	<code>\@glsxtr@current@style</code>	50, 381
<code>\@glsstyle@treename</code>	362	<code>\@glsxtr@currentunitcount</code>	97, 98, 100
<code>\@glsstyle@treenamegroup</code>	363	<code>\@glsxtr@currunitcount</code>	99, 101
<code>\@glsstyle@treenamehypergroup</code> ...	363	<code>\@glsxtr@declareoption</code>	5, 15, 17, 20
<code>\@glstarget</code>	80, 111	<code>\@glsxtr@defaultnoglossarywarning</code> ..	20
<code>\@glstext@</code>	81	<code>\@glsxtr@defaultnumberformat</code>	
<code>\@glswidestname</code>	366, 373	7, 9, 58, 60, 76, 164, 167
<code>\@glsxtr</code>	47, 49	<code>\@glsxtr@defpostpunc</code>	15, 16, 23
<code>\@glsxtr@do@wrglossary</code>	107	<code>\@glsxtr@deprecated@abbrstyle</code>	
<code>\@glsxtr@abbreviationsdef</code>	17, 24	226, 227, 229,
<code>\@glsxtr@accessdisplay</code>	164–166	231, 240, 242, 243, 245, 258, 261, 265, 267
<code>\@glsxtr@activate@initialtagging</code> ..		<code>\@glsxtr@disabledflycommand</code>	49
.....	172, 173	<code>\@glsxtr@display@loc</code>	116
<code>\@glsxtr@addunitcounter</code>	97	<code>\@glsxtr@do@windex</code>	77
<code>\@glsxtr@addunused</code>	44	<code>\@glsxtr@do@gl:disablehyperinlist</code> ..	75
<code>\@glsxtr@addunusedxrefs</code>	44	<code>\@glsxtr@do@inc@linkcount</code>	142
<code>\@glsxtr@attrval</code>		<code>\@glsxtr@do@record@wrglossary</code>	8, 13
.....	59, 159, 160, 162, 163, 165, 167	<code>\@glsxtr@do@redef@for@gl:sentries</code>	7
<code>\@glsxtr@autoindex@at</code>	167–169	<code>\@glsxtr@do@style</code>	21, 313
<code>\@glsxtr@autoindex@doextra@esc</code>	167	<code>\@glsxtr@do@titlecaps@warn</code>	
<code>\@glsxtr@autoindex@encap</code>	167–169	159–162, 165, 172, 173
<code>\@glsxtr@autoindex@esc</code>	168, 170, 171	<code>\@glsxtr@doabbreviationsdef</code>	17
<code>\@glsxtr@autoindex@escat</code>	168, 169	<code>\@glsxtr@doaccsupp</code>	20, 22
<code>\@glsxtr@autoindex@escencap</code> ...	168, 169	<code>\@glsxtr@docdefval</code>	14, 46
<code>\@glsxtr@autoindex@esclevel</code> ...	168, 169	<code>\@glsxtr@docounterrecord</code>	11
<code>\@glsxtr@autoindex@escquote</code> ...	168, 170	<code>\@glsxtr@doglossary</code>	128, 129
<code>\@glsxtr@autoindex@level</code>	168, 169	<code>\@glsxtr@doiflabelinlist</code>	130

\@glxtr@doloctag	52, 53	\@glxtr@noidx@numberlistloop	108
\@glxtr@dorecord	8, 9	\@glxtr@noop@recordcounter	10, 13
\@glxtr@dorecordnodefer	8, 9	\@glxtr@nopostpunc	110
\@glxtr@dosee@alsoindex@glossary ..	13	\@glxtr@nopostpunc@postdesc ..	110, 111
\@glxtr@doseeglossary	13, 23	\@glxtr@notfoundinlist	177
\@glxtr@dostylewarn	197	\@glxtr@op@recordcounter	13
\@glxtr@enabletagging	171	\@glxtr@optlist	49
\@glxtr@end@	47	\@glxtr@org@starttoc	294
\@glxtr@endescspch	168–171	\@glxtr@org@GLS@	56
\@glxtr@entrycount@org@localreset ..	92	\@glxtr@org@GLSpl@	56
\@glxtr@entrycount@org@localunset ..	91	\@glxtr@org@Gls@	55
\@glxtr@entrycount@org@reset	92	\@glxtr@org@Glspl@	55
\@glxtr@entrycount@org@unset	91	\@glxtr@org@Glsxtrtitlefirst ..	295, 296
\@glxtr@entryunitcount@org@localreset	100	\@glxtr@org@Glsxtrtitlefirstplural	295, 296
\@glxtr@entryunitcount@org@localunset	99, 100	\@glxtr@org@Glsxtrtitlefull ..	295, 296
\@glxtr@entryunitcount@org@reset ..	100	\@glxtr@org@Glsxtrtitlefullpl ..	295, 296
\@glxtr@entryunitcount@org@unset ..	99	\@glxtr@org@Glsxtrtitlelong ..	295, 296
\@glxtr@err@undefaction	7, 13	\@glxtr@org@Glsxtrtitlelongpl ..	295, 296
\@glxtr@field@linkdefs	55	\@glxtr@org@Glsxtrtitlename ..	295, 296
\@glxtr@format@overridefalse	166	\@glxtr@org@Glsxtrtitleplural ..	295, 296
\@glxtr@format@overridefalse ..	166, 167	\@glxtr@org@Glsxtrtitleshort ..	295, 296
\@glxtr@foundinlist	177	\@glxtr@org@Glsxtrtitleshortpl ..	295, 296
\@glxtr@full	184	\@glxtr@org@Glsxtrtitletext ..	295, 296
\@glxtr@fullpl	186	\@glxtr@org@MakeUppercase	295, 296
\@glxtr@gettype	108	\@glxtr@org@checkfirsthyper ...	74, 105
\@glxtr@glossdescfont	159, 160	\@glxtr@org@delimN	52, 53
\@glxtr@glossnamefont	160–166	\@glxtr@org@delimR	52, 53
\@glxtr@gobbleto@endescspch	170	\@glxtr@org@doseeglossary	23, 106
\@glxtr@groupheading	129, 131	\@glxtr@org@gloautosee	24
\@glxtr@idx@displaynumberlist	107	\@glxtr@org@gls@	55
\@glxtr@idx@entrynumberlist	108	\@glxtr@org@glsdohypertarget ..	111, 112
\@glxtr@ifcsstart	47	\@glxtr@org@glsignore	52, 53
\@glxtr@ifpunctoken	177	\@glxtr@org@glspl@	55
\@glxtr@ifunitcounter	96	\@glxtr@org@glsxtrtitlefirst ..	295, 296
\@glxtr@insert@dots	179	\@glxtr@org@glsxtrtitlefirstplural	295, 296
\@glxtr@insert@dots@next	179	\@glxtr@org@glsxtrtitlefull ..	295, 296
\@glxtr@insertdots	181	\@glxtr@org@glsxtrtitlefullpl ..	295, 296
\@glxtr@label	30, 44, 142, 158	\@glxtr@org@glsxtrtitlelong ..	295, 296
\@glxtr@loadstyles	347, 348	\@glxtr@org@glsxtrtitlelongpl ..	295, 296
\@glxtr@longnewglossaryentry	33	\@glxtr@org@glsxtrtitlename ..	295, 296
\@glxtr@mark@wordseps	179	\@glxtr@org@glsxtrtitleorpdforheading	295, 296
\@glxtr@mark@wordseps@next	180	\@glxtr@org@glsxtrtitleplural ..	295, 296
\@glxtr@markwordseps	180–182	\@glxtr@org@glsxtrtitleshort ..	295, 296
\@glxtr@mixed@assign@sortkey	108	\@glxtr@org@glsxtrtitleshortpl ..	295, 296
\@glxtr@noidx@displaynumberlist ..	107	\@glxtr@org@glsxtrtitletext ..	295, 296
\@glxtr@noidx@do	131	\@glxtr@org@makeglossaries	105
\@glxtr@noidx@entrynumberlist	108		

<code>\@glxtr@org@markboth</code>	293, 294	<code>\@glxtr@taggingcs</code>	172
<code>\@glxtr@org@markright</code>	293, 294	<code>\@glxtr@textformat</code>	59
<code>\@glxtr@org@newacronymstyle</code>	104	<code>\@glxtr@theHvalue</code> ..	8, 9, 57, 58, 60, 136, 137
<code>\@glxtr@org@postdescription</code> ..	111, 173	<code>\@glxtr@thevalue</code> ..	8, 9, 57, 58, 60, 136, 137
<code>\@glxtr@org@see@noindex</code>	123	<code>\@glxtr@thisloctag</code>	53
<code>\@glxtr@org@setacronymstyle</code>	104	<code>\@glxtr@titlelabel</code>	114, 115, 131
<code>\@glxtr@org@theHvalue</code>	8, 9	<code>\@glxtr@tmp</code>	21, 117
<code>\@glxtr@orgprefix</code>	10	<code>\@glxtr@type</code>	158
<code>\@glxtr@orgprintglossary</code>	49, 111	<code>\@glxtr@unitcountlist</code>	97
<code>\@glxtr@orgwarndep</code>	178	<code>\@glxtr@unsrt@getgroupitle</code>	128
<code>\@glxtr@p@acrlong@</code>	81	<code>\@glxtr@usesee</code>	39
<code>\@glxtr@p@acrlongpl@</code>	81	<code>\@glxtr@warn@onexistsordo</code>	7, 13
<code>\@glxtr@p@acrshort@</code>	81	<code>\@glxtr@warn@undefaction</code>	7, 13
<code>\@glxtr@p@acrshortpl@</code>	81	<code>\@glxtr@rdialecthook</code>	313, 345
<code>\@glxtr@p@long@</code>	81	<code>\@glxtr@docdeffalse</code>	45
<code>\@glxtr@p@longpl@</code>	81	<code>\@glxtr@entryfmt</code>	28
<code>\@glxtr@p@plural@</code>	81	<code>\@glxtr@fmt</code>	27
<code>\@glxtr@p@short@</code>	81	<code>\@glxtr@glossentry</code>	126
<code>\@glxtr@p@shortpl@</code>	81	<code>\@glxtr@glossentryother</code>	127
<code>\@glxtr@p@text@</code>	81	<code>\@glxtr@hypernameprefix</code>	111, 112, 130
<code>\@glxtr@pagetag</code>	52, 53	<code>\@glxtr@trifhasfield</code>	30
<code>\@glxtr@pagetag</code>	52, 53	<code>\@glxtr@trifhyphenstart</code>	276
<code>\@glxtr@prevunitcount</code>	99	<code>\@glxtr@indexaliased</code>	75
<code>\@glxtr@printglossopts</code>	49, 108, 111	<code>\@glxtr@indexcrossrefsfalse</code>	15
<code>\@glxtr@printunsrtglossaryskipentry</code>	129	<code>\@glxtr@indexcrossrefstrue</code>	15
<code>\@glxtr@provide@addstoragekey</code>	27	<code>\@glxtr@inmark</code>	293, 294
<code>\@glxtr@provide@storagekey</code>	26	<code>\@glxtr@long</code>	81, 189
<code>\@glxtr@record</code>	13, 55, 56, 60	<code>\@glxtr@longpl</code>	81, 192, 193
<code>\@glxtr@record@setting</code>	8, 9, 12, 41, 45, 46, 105, 106	<code>\@glxtr@newgls</code>	134, 135
<code>\@glxtr@record@setting@alsoindex</code> ..	8, 9, 41, 106	<code>\@glxtr@newgls@inner</code>	133
<code>\@glxtr@record@setting@off</code>	45	<code>\@glxtr@newgls@innercsname</code>	133, 134
<code>\@glxtr@record@setting@only</code>	105	<code>\@glxtr@notinmark</code>	293, 294
<code>\@glxtr@recordsee</code>	13, 23, 41	<code>\@glxtr@p</code>	86, 87
<code>\@glxtr@redef@forglsentries</code>	7, 24	<code>\@glxtr@p@opt</code>	84
<code>\@glxtr@redefstyles</code>	21, 313	<code>\@glxtr@pl</code>	48, 49
<code>\@glxtr@reg@glosslist</code>	106–109, 112	<code>\@glxtr@postloctag</code>	51, 52, 54
<code>\@glxtr@restore@postpunc</code>	110, 111	<code>\@glxtr@preloctag</code>	51, 52, 54
<code>\@glxtr@rglstrigger@record</code> ...	137–139	<code>\@glxtr@setaliasnoindex</code>	75, 76
<code>\@glxtr@s@longnewglossaryentry</code>	33	<code>\@glxtr@short</code>	81, 188
<code>\@glxtr@savepreloctag</code>	52, 53	<code>\@glxtr@shortpl</code>	81, 191
<code>\@glxtr@setentrycountunsetattr</code>	90	<code>\@glxtr@undeftag</code>	6, 25
<code>\@glxtr@setentryunitcountunsetattr</code>	102	<code>\@glxtr@wrglossmark</code>	22
<code>\@glxtr@setupshortcuts</code>	19, 20, 24	<code>\@gobble</code> ..	7, 13, 15, 61, 129, 130, 179, 384–386
<code>\@glxtr@shortcutsval</code>	19, 125	<code>\@gobbletwo</code>	178
<code>\@glxtr@swaptwo</code>	178	<code>\@ifnextchar</code>	77
<code>\@glxtr@tag</code>	172	<code>\@ifpackageloaded</code>	5, 16, 125, 142, 158, 160, 163, 164, 166, 313, 314, 345
		<code>\@ifstar</code> ..	26, 27, 30, 33, 34, 36, 46, 77, 128, 171
		<code>\@ifundefined</code>	312

<code>\ACFP</code>	18	<code>\AS</code>	17
<code>\Acfp</code>	18	<code>\As</code>	17
<code>\acfp</code>	18	<code>\as</code>	17
<code>\ACL</code>	18	<code>\ASP</code>	17
<code>\Acl</code>	18	<code>\Asp</code>	17
<code>\acl</code>	18	<code>\asp</code>	17
<code>\ACLP</code>	18	<code>\AtBeginDocument</code>	22, 25, 50, 51, 125
<code>\Aclp</code>	18	<code>\AtEndDocument</code>	44, 92, 100, 118, 119
<code>\aclp</code>	18		
<code>\ACP</code>	18	B	
<code>\Acp</code>	18	babel package	167, 169, 176
<code>\acp</code>	18	<code>\begin</code>	116, 120, 121, 128, 349, 351–358, 377–380, 384
<code>\ACRfullfmt</code>	103	<code>\begingroup</code>	8, 9, 27, 28, 76, 126–128, 141
<code>\Acrfullfmt</code>	103	<code>\bgroup</code>	33, 34, 109
<code>\acrfullfmt</code>	103	<code>\bib2gls</code>	28, 131, 136, 137, 312, 313
<code>\ACRfullplfmt</code>	104		
<code>\Acrfullplfmt</code>	103	C	
<code>\acrfullplfmt</code>	103	<code>\catcode</code>	123
<code>\acronymentry</code>	103	category attributes:	
<code>\acronymfont</code>	68–72, 83, 104	apospplural	181
<code>\acronymname</code>	16	discardperiod	175
<code>\acronymsort</code>	103	entrycount	89–92, 102
<code>\acronymtype</code>	17, 103, 104	firstuc	162
<code>\acrpluralsuffix</code>	103, 124	glossdesc	158
<code>\ACS</code>	18	glossdescfont	159
<code>\Acs</code>	18	glossname	160
<code>\acs</code>	18	glossnamefont	160, 163
<code>\ACSP</code>	18	headuc	296
<code>\Acsp</code>	18	indexname	167
<code>\acsp</code>	18	indexonlyfirst	76
<code>\actualchar</code>	170	insertdots	181
<code>\addtolength</code>	374	linkcount	140
<code>\advance</code>	92, 101, 123, 133	linkcountmaster	141
<code>\AF</code>	17	markshortwords	181
<code>\Af</code>	17	markwords	180, 181, 276, 277, 285
<code>\af</code>	17	nohyper	75
<code>\AFP</code>	17	nohyperfirst	61–63
<code>\Afp</code>	17	noshortplural	181
<code>\afp</code>	17	regular	54, 96, 200–205, 207, 210, 212, 213, 215, 216, 219, 220, 222, 223, 225, 227–229, 233–237, 240–242, 244, 247–253, 255, 257, 259, 261–263, 265, 268, 274, 276, 278, 279, 281, 286, 291, 292
<code>\AL</code>	17	textformat	59
<code>\Al</code>	17	<code>\cdot</code>	22
<code>\al</code>	17	<code>\centering</code>	383
<code>\ALP</code>	17	<code>\cGLS</code>	17, 18, 90, 102
<code>\Alp</code>	17	<code>\cGls</code>	17, 18, 90, 102
<code>\alp</code>	17	<code>\cglS</code>	17, 18, 90, 102
<code>\AnyTrackedLanguages</code>	313, 345		
<code>\appto</code>	11, 12, 21, 26, 39, 41–43, 72, 77, 91, 99, 129, 130, 166, 176, 179, 180, 347		
<code>\arabic</code>	386		

<code>\cGLSformat</code>	94	<code>\def</code>	9–13, 23, 25, 27, 33–
<code>\cGLSformat</code>	94		36, 39, 41, 42, 44, 47–49, 51, 54–58, 60–
<code>\cGLSformat</code>	93, 95		72, 78, 80–84, 93–96, 103, 107–109, 111,
<code>\cGLSpl</code>	17, 18, 90, 102		112, 114–118, 128, 131, 133, 136–139,
<code>\cGLspl</code>	17, 18, 90, 102		168–173, 177–181, 185–194, 197, 276,
<code>\cGLspl</code>	17, 18, 90, 102		277, 279, 282, 285, 287, 374, 375, 379, 380
<code>\cGLSplformat</code>	94	<code>\defglentryfmt</code>	34–37
<code>\cGLsplformat</code>	94	<code>\define@boolkey</code>	14, 15, 57, 75
<code>\cGLsplformat</code>	93, 96	<code>\define@choicekey</code>	
<code>\changes</code>	299		7, 12, 14, 15, 19, 20, 22, 57, 111
<code>\char</code>	114	<code>\define@key</code>	
<code>\columnwidth</code>	50, 51		11, 12, 16, 21, 26, 42, 57, 60, 72, 111, 178
<code>\count@</code>	92, 93, 101	<code>\DefineAcronymSynonyms</code>	19
<code>\csappto</code>	32	<code>\delimN</code>	52, 53
<code>\csdef</code>	26, 31–33, 72–74, 92, 97, 98,	<code>\delimR</code>	52, 53
	100, 153, 197–199, 207, 229, 244, 265,	<code>\detokenize</code>	47
	269, 271, 273, 283, 284, 288, 290, 348, 365	<code>\dimen@</code>	105, 365–373
<code>\cseappto</code>	37	<code>\dimen@i</code>	367–369
<code>\csedef</code>	31, 98, 115	<code>\dimen@ii</code>	365–369
<code>\csgdef</code>	31,	<code>\dimexpr</code>	50, 51, 364
	34–37, 45, 52, 92, 97, 100, 101, 364, 365, 387	<code>\disable@keys</code>	17, 25, 46, 123
<code>\cslet</code>	31, 33, 34, 110	<code>\do</code>	6, 21, 30, 44, 91, 102,
<code>\csletcs</code>	31, 199		106, 109, 115, 129, 136, 142, 158, 164, 172
<code>\csname</code>	6, 27, 35, 36, 41,	<code>\do@gl@link@checkfirsthyper</code>	
	45, 50, 54, 56, 58, 60, 68–74, 76, 84, 98,		28, 55, 56, 58, 68–72, 185–194
	106, 107, 115, 118, 119, 121, 122, 129,	<code>\do@gl@disablehyperinlist</code>	58, 75
	133–136, 141, 158, 178, 185–194, 200, 373	<code>doc package</code>	170
<code>\cspreto</code>	33	<code>\dolistcsloop</code>	29
<code>\csuse</code>	28, 35, 43, 52, 73, 74, 86–88, 96–101,	<code>\DTLifinlist</code>	106–108, 112
	109, 114–116, 126, 127, 130–132, 153,	<code>\DTLifint</code>	114
	164, 174, 175, 198, 199, 365, 366, 368, 369		
<code>\csxdef</code>	39, 42, 98, 100, 115		E
<code>\currentglossary</code>	110, 126, 127, 386	<code>\eappto</code>	11, 21, 34–37, 129, 131, 167, 347
<code>\CurrentOption</code>	22, 347, 348	<code>\edef</code>	6, 8–10, 34–37, 40,
<code>\CurrentTrackedLanguage</code>	345		42, 43, 45, 58–60, 74, 76, 78, 80, 96–98,
<code>\CurrentTrackedLanguageTag</code>	124		100, 106, 107, 112, 114, 117–119, 123,
<code>\CurrentTrackedScript</code>	345		126, 127, 136, 141, 159, 160, 162–165,
<code>\CurrentTrackedTag</code>	312, 345		168, 170, 171, 178, 345, 368, 369, 384, 385
<code>\CustomAbbreviationFields</code>	182,	<code>\eglssetwidest</code>	366–373
	200, 202–205, 207, 209, 211, 214, 216–	<code>\egroup</code>	33, 34, 110
	222, 224, 228, 229, 232–236, 239, 242,	<code>\else</code>	8–11, 14, 15, 17, 19,
	243, 246, 247, 249–254, 256, 258, 261,		20, 23, 28, 32, 45, 47, 52, 54, 56, 59, 76,
	263, 265, 268, 269, 271, 273–275, 277–		77, 93, 105, 106, 108–111, 114, 116, 117,
	279, 281, 282, 284–286, 288, 289, 291, 292		120, 122, 123, 136, 137, 166–168, 170,
			171, 177, 179–181, 188–194, 196, 197,
			201, 203, 204, 206–215, 218, 220–248,
			250–266, 268–270, 272, 273, 275–277,
			279, 282, 284, 285, 287, 289, 291, 292,
			349, 351–361, 363, 374, 375, 381, 383, 385
		<code>\emph</code>	245, 246
D			
<code>\DeclareAcronymList</code>	103		
<code>\DeclareOption</code>	5, 347		
<code>\DeclareOptionX</code>	5, 22		

<code>\empty</code>	116, 117	<code>\futurelet</code>	177
<code>\encapchar</code>	170	G	
<code>\end</code>	116,	<code>\gdef</code>	53, 169, 170
	120, 121, 129, 349, 351–358, 377–380, 384	<code>\Genacrfullformat</code>	103, 104
<code>\end@glxstr@display@loc</code>	116	<code>\genacrfullformat</code>	103, 104
<code>\endcsname</code>	6, 27, 35, 36, 41,	<code>\GenericAcronymFields</code>	103
	45, 50, 54, 56, 58, 60, 68–74, 76, 84, 98,	<code>\Genplacrfullformat</code>	103, 104
	106, 107, 115, 118, 119, 121, 122, 129,	<code>\genplacrfullformat</code>	103, 104
	133–136, 141, 158, 178, 185–194, 200, 373	<code>\glo@grabfirst</code>	131
<code>\endgroup</code>	8, 10, 28, 76, 126–128, 141	<code>\glo@name</code>	161, 162, 166
<code>\ensuremath</code>	22	<code>\gloaliaslabel</code>	79
entry categories:		<code>\global</code>	10, 33, 34, 110, 132
abbreviation	194	<code>\glolinkprefix</code>	59, 79, 80, 125
general	153, 155	glossaries package	13, 23, 24, 39–41, 43, 109, 348
index	156	glossaries-accsupp package	20, 22, 142
<code>\epreto</code>	167	glossaries-extra package	2, 345
<code>\equal</code>	122	glossaries-extra-bib2gls package	13, 25, 313, 345
etoolbox package	5	glossaries-extra-stylemods package	20, 174, 313
<code>\expandafter</code>	22, 27,	glossaries-stylemods package	382
	28, 30, 39, 40, 44, 47–49, 73, 74, 77, 78,	glossaries.sty package	34
	84, 95–97, 106–108, 112, 115, 117, 128,	<code>\GlossariesExtraWarning</code>	6,
	129, 131, 133, 134, 140, 158, 161, 162,		15, 32, 33, 47, 49, 59, 104, 107, 117, 120,
	164, 166, 167, 170, 177, 180–182, 276, 384		123, 128, 159, 160, 162, 163, 165, 172, 199
<code>\expandonce</code>	103, 132, 168, 202, 279	<code>\GlossariesExtraWarningNoLine</code>	15, 93, 101
F		<code>\GlossariesWarning</code> ..	52, 107, 109, 113, 197
<code>\fi</code>	7–11, 14, 15, 17,	<code>\GlossariesWarningNoLine</code>	107, 119
	19, 20, 22–24, 28, 32, 39, 41, 42, 44, 46,	glossary styles:	
	47, 50–52, 54, 56, 57, 59, 76, 77, 93, 101,	altlist	349
	102, 105, 108–111, 114, 116, 117, 119,	altlistgroup	350
	120, 122, 123, 125, 136, 137, 166–169,	altlisthypergroup	351
	171, 177, 179, 180, 182, 188–194, 196,	almtree	363, 364, 374
	197, 201, 203, 204, 206–215, 218, 220–	almtreegroup	375
	248, 250–266, 268–270, 272, 273, 275–	almtreehypergroup	375
	277, 279, 282, 284, 285, 287, 289, 291,	index	359
	292, 349, 351–363, 365–375, 381, 383, 386	indexgroup	360
first use	388	indexhypergroup	360
flag	388	inline	359
text	388	list	349
<code>\firstacronymfont</code>	104, 105	listdotted	348
fontspec package	125	listdottedstyle	349
<code>\footnote</code>	205	listgroup	350
<code>\forall glossaries</code>		listhypergroup	350
	44, 128, 156, 158, 366, 367, 369–373	mcolalmtree	379
<code>\forall glsentries</code>	92, 101	mcolalmtreegroup	379
<code>\ForEachTrackedDialect</code>	313, 345	mcolalmtreehypergroup	380
<code>\for glsentries</code>	6, 44, 156, 158, 366–373	mcolalmtreespannav	380
<code>\forlistcsloop</code>	29, 101, 116	mcolindexgroup	376
<code>\forlistloop</code>	112, 113, 173	mcolindexhypergroup	376

mcolindexspannav	376	\gls@level	132
mcoltreegroup	377	\gls@noidxglossary	107
mcoltreehypergroup	377	\gls@org@glossaryentryfield	110
mcoltreenonamegroup	378	\gls@org@glossarysubentryfield	110
mcoltreenonamehypergroup	378	\gls@save@numberlist	51, 52, 54
mcoltreenonamespannav	379	\gls@set@xr@key	41, 42
mcoltreespannav	377	\gls@tmplen	366–375
sublistdotted	349	\gls@type	107
tree	361	\glsabbrvdefaultfont ...	184, 201, 203, 206, 208, 209, 211, 214, 267, 277, 280, 290
treegroup	361	\glsabbrvemfont	245–254, 256, 258, 260, 262–266
treehypergroup	362	\glsabbrvfont	82, 83, 104, 184, 188, 189, 191, 192, 194, 196, 197, 200–209, 211, 214, 216, 218, 219, 221, 222, 225, 226, 228, 230, 232, 234, 235, 237, 239, 240, 242, 244, 246, 248, 250, 251, 253, 254, 256, 258, 260, 262, 264, 266, 268, 269, 272, 274–276, 278, 280, 282, 283, 286, 288, 291
treenoname	362	\glsabbrvhyphenfont	277, 278, 282–286, 288, 289
treenonamegroup	363	\glsabbrvonlyfont	290–292
treenonamehypergroup	363	\glsabbrvscfont .	217–222, 225, 226, 228–230
glossary-bookindex package	348	\glsabbrvsmfont	231–235, 237, 239, 240, 242, 244
glossary-hypernav package	78	\glsabbrvuserfont	267–275
glossary-long package	353	\GLSaccessdesc	64
glossary-longbooktabs package	353	\Glsaccessdesc	64, 159, 171
\glossaryentrynumbers	54, 109, 110, 132, 133	\glsaccessdesc	64, 159, 175
\glossaryheader	116, 128, 349–358, 360–363, 374–378, 380, 384	\GLSaccessdescplural	65
\glossaryname	109	\Glsaccessdescplural	65
\glossarypostamble	116, 129, 130	\glsaccessdescplural	65
\glossarypreamble	115, 128	\GLSaccessfirst	62
\glossarysection ...	115, 116, 121, 128, 130	\Glsaccessfirst	62
\glossarytitle	35, 36, 109, 110, 115, 116, 121, 128	\glsaccessfirst	62
\glossarytoctitle	35, 36, 109, 115, 116, 121, 128	\GLSaccessfirstplural	63
\glossentry	110, 132, 348, 349, 351–358, 360–362, 374, 384	\Glsaccessfirstplural	63
\glossentrydesc	348–358, 360–364	\glsaccessfirstplural	63
\glossentryname	126, 348–358, 360–362, 374, 375, 382	\Glsaccesslong	71, 183, 190, 201, 210, 214, 215, 218, 224–227, 232, 238–241, 246, 248, 256–260, 262, 269, 270, 278, 280, 281, 283, 284, 286, 291, 292
\glossentrynameother	127	\glsaccesslong	70, 71, 183, 189, 190, 201, 203, 204, 206, 208, 209, 211, 213–215, 218, 220, 221, 223–229, 231, 232, 234–243, 245, 246, 248, 250, 252–266, 268, 270, 272, 273, 275, 278, 280, 281, 283, 284, 286, 291
\glossentrysymbol	352–356, 358, 360–362, 364		
\glossxtrsetpopts	174		
\GLS	90, 102, 135		
\Gls	48, 90, 102, 135		
\gls	32, 48, 49, 90, 102, 107, 120, 135		
\gls@assign@desc	33, 34		
\gls@assign@field	11, 12, 26, 72		
\gls@checkseeallowed	46, 106		
\gls@codepage	119		
\gls@defdocnewglossaryentry	91, 99		
\gls@defglossaryentry	33, 34, 48, 49		
\gls@dotocitle	109, 110		
\gls@glossary	41		
\gls@grplabel	78		

<code>\Glsaccesslongpl</code>	72, 183, 193, 201, 210, 214, 215, 218, 224, 225, 227, 232, 238–241, 247, 248, 256–262, 269, 270, 278, 280, 281, 284, 286, 291, 292	<code>\glsadd options</code>	
<code>\glsaccesslongpl</code>	71, 72, 183, 193, 194, 201, 203, 204, 206–209, 211–215, 218, 220–229, 231, 232, 234, 236–243, 245, 246, 248, 250, 252, 253, 255–268, 270, 272, 273, 275, 278, 280, 281, 283, 284, 286, 291, 292	<code>theHvalue</code>	9
<code>\GLSaccessname</code>	64	<code>thevalue</code>	9
<code>\Glsaccessname</code>	64	<code>\glsaddstoragekey</code>	43, 153
<code>\glsaccessname</code>	40, 64	<code>\glsbackslash</code>	47
<code>\GLSaccessplural</code>	63	<code>\glscapscase</code>	56, 60–74, 185–196
<code>\Glsaccessplural</code>	62	<code>\glscategory</code> 54, 61, 74, 82, 83, 154, 155, 158–160, 162–165, 171, 174, 175, 185–189, 191, 192
<code>\glsaccessplural</code>	62	<code>\glscategorylabel</code> 74, 178, 180, 181, 207, 229, 244, 265, 269, 271, 273, 283, 284, 288, 290
<code>\Glsaccessshort</code>	69, 188, 197, 203, 206, 208, 211, 213, 220, 221, 223, 228– 231, 234, 236, 237, 243–245, 250, 252, 253, 255, 264–266, 272, 275, 283, 288, 289	<code>\glsclosebrace</code>	41, 121, 122
<code>\glsaccessshort</code>	68, 69, 183, 188, 189, 196, 201, 203, 206, 208–213, 215, 218, 220– 225, 227–232, 234–246, 248, 250, 251, 253–257, 259–262, 264, 266, 268–270, 272, 275, 278, 280, 283, 286, 288, 289, 292	<code>\glscurrententrylabel</code> 51–53, 110, 118, 126–130, 173, 174
<code>\Glsaccessshortpl</code>	70, 192, 197, 204, 206–208, 212, 213, 220, 222, 223, 229–231, 234, 236, 237, 243, 245, 250, 252, 253, 255, 264–266, 272, 273, 275, 283, 288, 289, 292	<code>\glscurrentfieldvalue</code>	28, 30, 32, 267
<code>\glsaccessshortpl</code>	69, 70, 183, 191, 192, 196, 201, 203, 206, 208–213, 215, 218, 220– 225, 227–234, 236–241, 243–248, 250, 252–257, 259–262, 264, 266, 269, 270, 272, 275, 278, 280, 283, 286, 288, 289, 292	<code>\glscustomtext</code> ..	55, 56, 68–72, 185–194, 196
<code>\GLSaccesssymbol</code>	65	<code>\glsdefaulttype</code> 6, 16, 32, 108, 109, 120, 128, 130
<code>\Glsaccesssymbol</code>	65, 171	<code>\glsdescriptionaccessdisplay</code>	146, 147, 159
<code>\glsaccesssymbol</code>	65, 171, 175	<code>\glsdescriptionpluralaccessdisplay</code>	147
<code>\GLSaccesssymbolplural</code>	66	<code>\glsdescwidth</code>	351–358
<code>\Glsaccesssymbolplural</code>	66	<code>\glsdetoklabel</code>	8, 9, 29–34, 37– 40, 44, 45, 47, 58, 60, 76, 79, 91, 92, 97– 101, 106, 110, 112, 113, 126, 127, 131, 132, 135, 136, 158, 161, 162, 166, 368, 369
<code>\glsaccesssymbolplural</code>	66	<code>\glsdisplaynumberlist</code>	107, 112
<code>\GLSaccessstext</code>	61	<code>\glsdohyperlink</code>	78, 80
<code>\Glsaccessstext</code>	61	<code>\glsdohypertarget</code>	80, 111
<code>\glsaccessstext</code>	40, 61	<code>\glsdoifexists</code> ..	14, 23, 31, 37, 39–41, 55, 56, 60, 68–72, 80, 106, 113, 126, 127, 185–194
<code>\glsacrshortcutstrue</code>	19, 20	<code>\glsdoifexistsordo</code>	27, 28, 56
<code>\glsacspacemax</code>	105	<code>\glsdoifexistsorwarn</code>	14, 158, 159, 171
<code>\glsadd</code>	28, 44, 120	<code>\glsdoifnoexists</code>	33
		<code>\glsdonohyperlink</code>	59, 80
		<code>\glsdosanitizesort</code>	108
		<code>\glsenableentrycount</code>	90, 93, 101
		<code>\glsenableentryunitcount</code>	92, 102
		<code>\glsentrycounter</code>	117
		<code>\glsentrycurrcount</code>	91, 92, 99
		<code>\Glsentrydesc</code>	146, 151, 160
		<code>\glsentrydesc</code>	146, 147, 151, 160
		<code>\Glsentrydescplural</code>	147, 151
		<code>\glsentrydescplural</code>	147, 151, 152
		<code>\Glsentryfirst</code>	96, 140, 144, 150
		<code>\glsentryfirst</code>	96, 139, 144, 150, 309
		<code>\Glsentryfirstplural</code>	96, 140, 145, 150
		<code>\glsentryfirstplural</code>	96, 140, 145, 150, 151, 309

<code>\glsentryfmt</code>	34–37	<code>\glsfieldxdef</code>	158
<code>\Glsentryfull</code>	104	<code>\glsfindwidesttoplevelname</code>	366
<code>\glsentryfull</code>	104	<code>\GLSfirst</code>	301
<code>\Glsentryfullpl</code>	104	<code>\Glsfirst</code>	301
<code>\glsentryfullpl</code>	104	<code>\glsfirst</code>	301
<code>\glsentryitem</code>	126, 127, 348, 349, 351–358, 360–362, 374, 385	<code>\glsfirstabbrvdefaultfont</code>	184, 201, 203, 206, 208, 209, 211, 214, 280
<code>\Glsentrylong</code>	83, 84, 96, 140, 149, 152	<code>\glsfirstabbrvmfont</code>	246–266
<code>\glsentrylong</code>	83, 84, 96, 139, 149, 152, 207, 230, 244, 265, 271, 273, 287, 310	<code>\glsfirstabbrvfont</code>	104, 183, 201–215, 218, 219, 221, 223, 225, 226, 228, 230, 232, 234, 235, 237, 239, 240, 242, 244, 246, 248, 250, 251, 253, 254, 256, 258, 260, 262, 264, 266, 268, 269, 272, 275, 278, 280, 283, 286, 288, 291
<code>\Glsentrylongpl</code>	83, 84, 96, 149, 152	<code>\glsfirstabbrvhyphenfont</code>	277, 278, 282, 283, 285–289
<code>\glsentrylongpl</code>	83, 84, 96, 149, 152, 310, 311	<code>\glsfirstabbrvonlyfont</code>	291, 292
<code>\Glsentrylongplural</code>	140	<code>\glsfirstabbrvscfont</code>	217–231
<code>\glsentrylongplural</code>	140	<code>\glsfirstabbrvsmfont</code>	232–245
<code>\Glsentryname</code>	142, 150, 161–164	<code>\glsfirstabbrvuserfont</code>	268–275
<code>\glsentryname</code>	126, 142, 143, 149, 150, 167, 307, 366–373, 387	<code>\glsfirstaccessdisplay</code>	144
<code>\glsentrynumberlist</code>	108, 114, 371–373	<code>\glsfirstlongdefaultfont</code>	201, 203, 209, 211, 214, 217–227, 232– 242, 246, 247, 249, 250, 253–257, 260, 261
<code>\Glsentryplural</code>	144, 150	<code>\glsfirstlongemfont</code>	247–249, 251, 252, 258, 259, 261–263
<code>\glsentryplural</code>	143, 144, 150, 308	<code>\glsfirstlongfont</code>	183, 201–204, 206, 208–216, 218, 220, 221, 223, 225, 226, 228, 230, 232, 234, 235, 237, 239, 240, 242, 244, 246, 248, 250, 251, 253, 254, 256, 258, 260, 262, 264, 266, 268, 269, 272, 275, 278, 280, 283, 286, 288, 291
<code>\glsentryprevcount</code>	91, 93, 99	<code>\glsfirstlongfootnotefont</code>	205–208, 228–231, 242–245, 263–267
<code>\glsentryprevmaxcount</code>	99	<code>\glsfirstlonghyphenfont</code>	277–288
<code>\glsentryprevtotalcount</code>	99	<code>\glsfirstlongonlyfont</code>	291, 292
<code>\Glsentryshort</code>	82, 83, 148, 152	<code>\glsfirstlonguserfont</code>	268–276
<code>\glsentryshort</code>	82, 83, 105, 147, 148, 152, 269, 271, 282, 306	<code>\GLSfirstplural</code>	301, 302
<code>\Glsentryshortpl</code>	82, 83, 148, 152	<code>\Glsfirstplural</code>	302
<code>\glsentryshortpl</code> ..	82, 83, 148, 152, 306, 307	<code>\glsfirstplural</code>	302
<code>\Glsentrysymbol</code>	145, 151	<code>\glsfirstpluralaccessdisplay</code> ..	144, 145
<code>\glsentrysymbol</code>	145, 146, 151, 370–372	<code>\glsforeachincategory</code>	197
<code>\Glsentrysymbolplural</code>	146, 151	<code>\glsgenentryfmt</code>	54
<code>\glsentrysymbolplural</code>	146, 151	<code>\glsgetattribute</code>	59, 79, 93, 97–99, 118, 136, 141, 159, 160, 162, 163, 165, 167
<code>\Glsentrytext</code>	143, 150	<code>\glsgetcategoryattribute</code>	154
<code>\glsentrytext</code>	80, 143, 150, 307, 308	<code>\glsgetgrouptitle</code>	350, 351, 360–363, 375–381
<code>\glsentrytype</code>	126, 127	<code>\glsgetwidestname</code>	364
<code>\Glsentryuseri</code>	66		
<code>\glsentryuseri</code>	66		
<code>\Glsentryuserii</code>	66		
<code>\glsentryuserii</code>	67		
<code>\Glsentryuseriii</code>	67		
<code>\glsentryuseriii</code>	67		
<code>\Glsentryuseriv</code>	67		
<code>\glsentryuseriv</code>	67		
<code>\Glsentryuserv</code>	67		
<code>\glsentryuserv</code>	68		
<code>\Glsentryuservi</code>	68		
<code>\glsentryuservi</code>	68		
<code>\glsextrapostnamehook</code>	164		
<code>\glsfieldfetch</code>	79		

<code>\glsgroupheading</code>		<code>noindex</code>	8, 75, 293
.... 132, 349–358, 360–363, 374–380, 386		<code>theHvalue</code>	58
<code>\glsgroupskip</code> 131, 349, 351–361, 363, 375, 386		<code>thevalue</code>	58, 133
<code>\glshasattribute</code>	59, 79,	<code>wrgloss</code>	8, 57
92, 93, 97, 98, 100, 101, 118, 136, 141,		<code>\glslinkcheckfirsthyperhook</code>	74
159, 160, 162, 163, 165, 167, 201–205,		<code>\glslinkpostsetkeys</code>	58, 136
207, 210, 212, 213, 216, 217, 219, 220,		<code>\glslinkpresetkeys</code>	58, 136
228, 230, 232–235, 242, 244, 246–249,		<code>\glslinkvar</code>	77, 78
251, 252, 259, 263–265, 268, 269, 271–		<code>\glslistchildpostlocation</code>	349
276, 278, 279, 281, 283, 285–288, 290–292		<code>\glslistchildprelocation</code>	349, 350
<code>\glshascategoryattribute</code>	154	<code>\glslistdottedwidth</code>	348
<code>\glshex</code> .. 316–321, 324–330, 332–335, 337–345		<code>\glslistgroupheaderfmt</code>	350, 351
<code>\glshyperlink</code>	80	<code>\glslistnavigationitem</code>	350, 351
<code>\glshypernavsep</code>	115	<code>\glslistprelocation</code>	349, 350
<code>\glshypernumber</code>	118, 166	<code>\glslocalunset</code>	56, 137
<code>\glsifattribute</code> .. 57, 61, 75, 77, 86, 140,		<code>\glslongaccessdisplay</code>	149
156, 159–162, 165, 166, 173, 176, 296–305		<code>\glslongdefaultfont</code> 184, 201, 203,	
<code>\glsifcategory</code>	156	205, 209, 211, 214, 218, 220, 221, 223–	
<code>\glsifcategoryattribute</code>		227, 232, 234, 235, 237, 239–241, 246,	
..... 74, 154, 155, 180, 181		250, 253, 254, 256, 257, 260, 267, 277, 290	
<code>\glsifnotregular</code>	61	<code>\glslongemfont</code> .. 245, 248, 251, 258, 261, 262	
<code>\glsifnotregularcategory</code>	155	<code>\glslongfont</code>	83, 184, 189, 190,
<code>\glsifplural</code>		193, 194, 201–204, 206, 208, 209, 211,	
56, 60, 62, 63, 65, 66, 68–72, 176, 185–195		214, 216, 218, 220, 221, 223, 225, 226,	
<code>\glsifregular</code>	54, 61, 96, 139, 140	228, 230, 232, 234, 235, 237, 239, 240,	
<code>\glsifregularcategory</code>	155	242, 244, 246, 248, 250, 251, 253, 254,	
<code>\glsifusetranslator</code>	35	256, 258, 260, 262, 264, 266, 268, 269,	
<code>\glsignore</code>	52, 53	272, 275, 278, 280, 283, 286, 288, 291, 292	
<code>\glsinlinedescformat</code>	359	<code>\glslongfootnotefont</code>	
<code>\glsinlinesubdescformat</code>	359	205, 206, 208, 228, 230, 242, 244, 264, 266	
<code>\glsinsert</code>	56,	<code>\glslonghyphenfont</code> . 277–283, 285, 286, 288	
60, 68–72, 185–196, 276, 283, 284, 288, 290		<code>\glslongonlyfont</code>	290, 291
<code>\glskeylisttok</code>	103, 180, 182	<code>\glslongpltok</code>	
<code>\glslabel</code>	8, 9, 27, 37, 182, 201–205, 214, 216–220, 224,	
40, 54, 57–59, 74–76, 79, 80, 105, 136,		228, 232, 233, 235, 239, 242, 246–252,	
140, 141, 174, 175, 194–196, 207, 230,		256, 258, 261, 263, 268, 269, 271, 273,	
244, 265, 269, 271, 273, 283, 284, 288, 290		274, 276–279, 281, 282, 284–286, 291, 292	
<code>\glslabeltok</code>	103, 180,	<code>\glslongpluralaccessdisplay</code>	149
182, 201–205, 207, 209–214, 216–222,		<code>\glslongtok</code>	
224, 228, 230, 232–235, 237, 239, 242,		. 103, 180, 182, 201–205, 207, 209, 211,	
244, 246–254, 256, 258, 259, 261, 263–		214, 216–222, 224, 228, 229, 232, 233,	
265, 268–279, 281, 283, 285–288, 290–292		235, 237, 239, 242, 244, 246–254, 256,	
<code>\glsletentryfield</code>	168	258, 261, 263, 265, 268–271, 273–275,	
<code>\glslink</code>	103, 104	277–279, 281, 282, 284–286, 288, 291, 292	
<code>\glslink options</code>		<code>\glslonguserfont</code>	268–275
counter	9	<code>\glsmcols</code>	377–380
format	166	<code>\GLSname</code>	298, 299
hyper	293	<code>\Glsname</code>	299
hyperoutside	57	<code>\glsname</code>	298

<code>\glsnameaccessdisplay</code> ..	142, 143, 161, 163	<code>\glssetcategoryattribute</code>	
<code>\glsnamefont</code>	160–165	..	91, 102, 105, 136, 142, 154, 155, 157, 172
<code>\glsnavhyperlink</code>	115	<code>\glssetnoexpandfield</code>	11, 12
<code>\glsnavhyperlinkname</code>	78	<code>\glssettoctitle</code>	109
<code>\glsnavhypertarget</code>		<code>\glsshortaccessdisplay</code>	147, 148
.....	350, 351, 361–363, 376–381	<code>\glsshortpltok</code>	
<code>\glsnavigation</code> ..	350, 351, 361–363, 375–380	182, 201–205, 207, 209, 211, 217, 219–	
<code>\glsnextpages</code>	110	222, 228, 229, 232, 233, 235, 237, 242,	
<code>\glsnoidxdisplayloc</code>	113	244, 246–254, 263, 265, 268, 269, 271,	
<code>\glsnoidxdisplayloclisthandler</code>	112	273–278, 282, 284–286, 288, 289, 291, 292	
<code>\glsnoidxloclist</code>	113, 132	<code>\glsshortpluralaccessdisplay</code>	148
<code>\glsnoidxnumberlistloophandler</code>	113	<code>\glsshorttok</code>	103, 180–182, 200–
<code>\glsnonnextpages</code>	110	205, 207, 209, 211, 216–222, 224, 228,	
<code>\glsnonnumberlistfalse</code>	51	229, 232–235, 237, 239, 242–244, 246,	
<code>\glsnonnumberlisttrue</code>	52	247, 249–254, 256, 258, 263, 265, 268–	
<code>\glsnopostdotfalse</code>	111	271, 273–278, 281, 282, 284–286, 288–292	
<code>\glsnopostdottrue</code>	110	<code>\glssubentryitem</code>	
<code>\glsnumberlistloop</code>	108	126, 127, 348–358, 360–362, 374, 385
<code>\glsnumlistlastsep</code>	112	<code>\glssymbolaccessdisplay</code>	145
<code>\glsnumlistsep</code>	112	<code>\glssymbolpluralaccessdisplay</code>	146
<code>\glsopenbrace</code>	41, 121, 122	<code>\glstarget</code>	126,
<code>\glsorder</code>	106	127, 348–358, 360–362, 374, 375, 385, 386	
<code>\glspagelistwidth</code>	352, 354, 356, 358	<code>\GLStext</code>	299
<code>\glspar</code>	130	<code>\Glstext</code>	299, 300
<code>\GLSpl</code>	90, 102, 135	<code>\glstext</code>	299
<code>\Glspl</code>	49, 90, 102, 135	<code>\glstextaccessdisplay</code>	143
<code>\glspl</code>	48, 90, 102, 135	<code>\glstextformat</code>	56, 59
<code>\GLSplural</code>	300	<code>\glstextup</code>	217
<code>\Glsplural</code>	300	<code>\glstreechildpredesc</code>	360, 361
<code>\glsplural</code>	300	<code>\glstreechildprelocation</code> ...	360, 361, 363
<code>\glspluralaccessdisplay</code>	143, 144	<code>\glstreegroupheaderfmt</code>	
<code>\glspluralsuffix</code>	124, 180, 184	360–363, 375–381, 383
<code>\glspostdescription</code>		<code>\glstreeindent</code>	361, 362, 373–375
..	15, 16, 110, 111, 173, 348–358, 360–364	<code>\glstreeitem</code>	359, 377, 385
<code>\glspostinline</code>	359	<code>\glstreenamebox</code>	374, 375
<code>\glspostlinkhook</code> ..	55, 56, 68–72, 84, 185–194	<code>\glstreenamefmt</code>	360–362, 364, 366–375
<code>\glsprestandardsort</code>	108	<code>\glstreenavigationfmt</code> ..	361–363, 375–380
<code>\glsresetentrylist</code>	116, 128	<code>\glstreepredesc</code>	360–362
<code>\glssee</code>	41, 43	<code>\glstreeprelocation</code>	359–362, 364
<code>\glsseeformat</code>	40, 45, 107, 113	<code>\glstreesubitem</code>	359, 385
<code>\glsseelist</code>	40	<code>\glstreesubsubitem</code>	360, 386
<code>\glssetabbrfmt</code> ..	54, 61, 82, 83, 158–160,	<code>\GlstLetField</code>	31
162, 163, 165, 171, 185–189, 191, 192, 194		<code>\glstype</code>	56, 58, 68–72, 136, 185–194
<code>\glssetattribute</code> ..	201–205, 207, 209–214,	<code>\glunset</code>	44, 56, 93, 94, 137
216–222, 224, 228, 230, 232–235, 237,		<code>\glswrite</code>	41, 106
239, 242, 244, 246–249, 251–254, 256,		<code>\glswriteentry</code>	8, 9
258, 259, 261, 263–265, 268, 269, 271–		<code>\Glsxtr</code>	49
276, 278, 279, 281, 283, 285–288, 290–292		<code>\glsxtr</code>	49
		<code>\glsxtr@@do@wrglossary</code>	8, 9, 11, 13

<code>\glxtr@addloclistfield</code>	13	<code>\glxtrabbrvpluralsuffix</code>	124,
<code>\glxtr@addunused</code>	44		184, 201, 203, 206, 208, 209, 211, 214,
<code>\glxtr@applyabbrvfmt</code>	194		217, 231, 245, 268, 277, 280, 283, 288, 290
<code>\glxtr@applyabbrvstyle</code>	178, 180, 197	<code>\glxtrabbrvtype</code>	16, 17, 182
<code>\glxtr@counterrecord</code>	126	<code>\glxtractivatenopost</code>	110
<code>\glxtr@do@alsoindex@wrglossary</code>	13	<code>\glxtraddallcrossrefs</code>	44
<code>\glxtr@doption</code>	5, 15, 16, 22, 24	<code>\glxtralalias</code>	76
<code>\glxtr@fields</code>	124	<code>\glxtrAltTreeIndent</code>	364
<code>\glxtr@headentry@p</code>	86, 87	<code>\glxtraltreeInit</code>	374, 379, 380
<code>\glxtr@hyperoutsidefalse</code>	57	<code>\glxtrAltTreePar</code>	364
<code>\glxtr@hyperoutsidetrue</code>	57	<code>\glxtrAltTreeSetHangIndent</code> ...	364, 374
<code>\glxtr@ifnextpunc</code>	177	<code>\glxtrAltTreeSetSubHangIndent</code>	375
<code>\glxtr@ifpunctoken</code>	177	<code>\glxtraltreeSubSymbolDescLocation</code>	375
<code>\glxtr@inc@linkcount</code>	58, 142	<code>\glxtraltreeSymbolDescLocation</code> ..	364, 374
<code>\glxtr@indexonly@saveentrycounter</code>		<code>\glxtrassignfieldfont</code>	61–68
.....	13, 25	<code>\glxtrautoindex</code>	167
<code>\glxtr@keylist</code>	47–49	<code>\glxtrautoindexassignsort</code>	167, 168
<code>\glxtr@label</code>	387	<code>\glxtrautoindexentry</code>	167
<code>\glxtr@langtag</code>	124	<code>\glxtrbookindexatendgroup</code>	385
<code>\glxtr@linkprefix</code>	124, 125	<code>\glxtrbookindexatsubendgroup</code>	385
<code>\glxtr@loaddialect</code>	313, 345	<code>\glxtrbookindexatsubsubendgroup</code> ..	385
<code>\glxtr@makeglossaries</code>	106	<code>\glxtrbookindexbetween</code>	385
<code>\glxtr@newabbreviation</code>	104, 180	<code>\glxtrbookindexbookmark</code>	386
<code>\glxtr@next</code>	177	<code>\glxtrbookindexcols</code>	384
<code>\glxtr@org@do@wrglossary</code>	25	<code>\glxtrbookindexcolspread</code>	384
<code>\glxtr@org@dohyperlink</code>	78	<code>\glxtrbookindexfirstmarkfmt</code>	387
<code>\glxtr@org@getgrouptitle</code>	114	<code>\glxtrbookindexformatheader</code>	386
<code>\glxtr@org@newignoredglossary</code>	34	<code>\glxtrbookindexgroupskip</code>	386
<code>\glxtr@orgmakenoidxglossaries</code>	45	<code>\glxtrbookindexlastmarkfmt</code>	387
<code>\glxtr@pluralsuffixes</code>	124	<code>\glxtrbookindexmulticolenv</code>	384
<code>\glxtr@process</code>	129	<code>\glxtrbookindexname</code>	382, 385
<code>\glxtr@provideignoredglossary</code>	36	<code>\glxtrbookindexparentchildsep</code>	382, 384
<code>\glxtr@punctlist</code>	176, 177	<code>\glxtrbookindexparentschildsep</code> ..	384, 385
<code>\glxtr@record</code>	10, 124	<code>\glxtrbookindexprelocation</code> ...	382, 385
<code>\glxtr@recordsee</code>	11	<code>\glxtrbookindexsubbetween</code>	385
<code>\glxtr@resource</code>	123, 124	<code>\glxtrbookindexsubname</code>	386
<code>\glxtr@s@newignoredglossary</code>	34	<code>\glxtrbookindexsubprelocation</code>	386
<code>\glxtr@s@provideignoredglossary</code> ...	36	<code>\glxtrbookindexsubsubbetween</code>	385
<code>\glxtr@saveentrycounter</code>	8, 9, 11, 76	<code>\glxtrbookindexthepage</code>	387
<code>\glxtr@setaccessdisplay</code>	165	<code>\glxtrcat</code>	48, 49
<code>\glxtr@setbookindexmark</code>	387	<code>\glxtrchecknohyperfirst</code>	62, 63
<code>\glxtr@setup@record</code>	12, 13, 24, 25	<code>\glxtrcombiningdiacriticIIIrules</code> .	317
<code>\glxtr@shortcutsval</code>	124, 125	<code>\glxtrcombiningdiacriticIIrules</code> ..	317
<code>\glxtr@texencoding</code>	124, 125	<code>\glxtrcombiningdiacriticIrules</code> ...	317
<code>\glxtr@usesee</code>	39	<code>\glxtrcombiningdiacriticIVrules</code> ..	317
<code>\glxtr@warnonexistsordo</code>	7, 13, 38	<code>\glxtrComputeTreeIndent</code>	374
<code>\glxtr@writefields</code>	123	<code>\glxtrComputeTreeSubIndent</code>	374
<code>\glxtrabbrvfootnote</code>			
.....	205–207, 228–230, 242–244, 263–265		

\glxtrcurrencyrules	320	223, 226–228, 230, 232, 234, 236, 237,
\Glsxtrdefaultsubsequentfmt ...	197, 198	240–242, 244, 246, 248, 250, 251, 253,
\glxtrdefaultsubsequentfmt ...	196, 198	255, 257, 259, 261, 262, 264, 266, 268,
\Glsxtrdefaultsubsequentplfmt .	197, 198	270, 272, 275, 278, 281, 283, 286, 289, 291
\glxtrdefaultsubsequentplfmt .	197, 198	\Glsxtrfullpl
\GlsXtrDefineAbbreviationShortcuts		17, 18, 304, 305
.....	19, 20	\Glsxtrfullpl
\GlsXtrDefineAcShortcuts	19, 20	17, 18, 304, 305
\GlsXtrDefineOtherShortcuts	19, 20	\Glsxtrfullplformat
\glxtrdetoklocation	135	. 183, 196, 198, 199, 201, 204, 206, 208,
\glxtrdiscardperiod	174	210, 212, 215, 218, 220, 222, 223, 226,
\glxtrdisplayendloc	116	227, 229, 230, 232, 234, 236, 238, 240,
\glxtrdisplayendlohook	117	242, 243, 245, 247, 248, 250, 252, 254,
\glxtrdisplaysingleloc	116, 117	255, 257, 259, 261, 263, 264, 266, 269,
\glxtrdisplaystartloc	116	270, 272, 275, 278, 281, 284, 286, 289, 291
\glxtrdoautoindexname	77, 164	\glxtrfullplformat
\glxtrdopostpunc	207, 230, 244, 265 195, 196, 198, 199, 201, 203,
\glxtrdowrglossaryhook	77	206, 208, 210, 212, 215, 218, 220, 222,
\glxtrremsuffix	246, 248, 250,	223, 226–228, 230, 232, 234, 236, 237,
251, 253, 254, 256, 258, 260, 261, 264, 266		240–242, 244, 246, 248, 250, 252, 254,
\GlsXtrEnableEntryCounting	102	255, 257, 259, 261, 263, 264, 266, 268,
\GlsXtrEnableEntryUnitCounting	90	270, 272, 275, 278, 281, 283, 286, 289, 291
\GlsXtrEnableOnTheFly	47, 49, 50	\glxtrfullsep
\glxtrendfor	30	. 183, 201–204, 206–213, 215, 217–225,
\glxtrfieldlistgadd	125	227, 229, 231–241, 243, 245–262, 264–
\glxtrfieldtitlecase	159–162, 166	267, 277, 278, 280, 282, 285–287, 291, 292
\glxtrfieldtitlecasecs	158	\glxtrrgenabbrvfmt
\glxtrfieldxifinlist	130	54
\glxtrfirstscfont	217	\glxtrrgeneralpuncIIrules
\glxtrfirstsmfont	231	320
\GlsXtrFmtDefaultOptions	28	\glxtrrgeneralpuncIrules
\glxtrfmtdisplay	28	320
\GlsXtrFmtField	28	\glxtrrgetgroupitle
\glxtrfootnotename		115, 386
.... 205, 207, 228, 229, 242, 243, 263, 265		\glxtrrgroupfield
\GlsXtrFormatLocationList 51, 54, 371–373		131
\Glsxtrfull	17, 18, 304, 305	\Glsxtrheadfirst
\Glsxtrfull	17, 18, 305	295
\glxtrfull	17, 18, 304	\glxtrheadfirst
\Glsxtrfullformat		295
.... 183, 196, 198, 199, 201, 203, 206,		\Glsxtrheadfirstplural
208, 210, 212, 215, 218, 220, 222, 223,		295
226–228, 230, 232, 234, 236, 237, 240,		\Glsxtrheadfull
241, 243, 244, 246, 248, 250, 252, 254,		296
255, 257, 259, 261, 263, 264, 266, 269,		\glxtrheadfullpl
270, 272, 275, 278, 281, 283, 286, 289, 291		296
\glxtrfullformat		\glxtrheadfullpl
..... 183, 196, 198, 199, 201, 203,		296
206, 208, 210, 212, 215, 218, 220, 222,		\Glsxtrheadlong
		296
		\glxtrheadlong
		295
		\Glsxtrheadlongpl
		296
		\glxtrheadlongpl
		295
		\Glsxtrheadname
		295
		\glxtrheadname
		126, 295
		\Glsxtrheadplural
		295
		\glxtrheadplural
		295
		\Glsxtrheadshort
		295
		\glxtrheadshort
		295
		\Glsxtrheadshortpl
		295

<code>\glxtrheadshortpl</code>	295	<code>\glxtrLatinA</code>	322–326
<code>\Glsxtrheadtext</code>	295	<code>\glxtrLatinAELigature</code>	324, 326
<code>\glxtrheadtext</code>	295	<code>\glxtrLatinE</code>	322–326
<code>\glxtrhyperlink</code>	79, 118	<code>\glxtrLatinEszettSs</code>	323–325, 327
<code>\glxtrhyphensuffix</code>	278, 286	<code>\glxtrLatinEszettSz</code>	323, 325
<code>\glxtrifcounttrigger</code>	93, 94	<code>\glxtrLatinEth</code>	322–326
<code>\glxtrifcustomdiscardperiod</code>	174	<code>\glxtrLatinH</code>	322–327
<code>\glxtrifemptyglossary</code>	116, 121, 128	<code>\glxtrLatinI</code>	322–327
<code>\glxtrifhasfield</code>	32, 75, 382	<code>\glxtrLatinInsularG</code>	326
<code>\glxtrifhyphenstart</code>	277, 279, 282, 285, 287	<code>\glxtrLatinK</code>	322–327
<code>\glxtrifindexing</code>	76	<code>\glxtrLatinL</code>	322–327
<code>\glxtrifinmark</code>	60, 86–89, 294–296	<code>\glxtrLatinM</code>	322–327
<code>\glxtrifnextpunc</code>	177, 178	<code>\glxtrLatinN</code>	322–327
<code>\glxtrifperiod</code>	174, 176	<code>\glxtrLatinO</code>	322–327
<code>\glxtrifrecordtrigger</code>	137–139	<code>\glxtrLatinOELigature</code>	324, 326, 327
<code>\glxtrifwasfirstuse</code>		<code>\glxtrLatinP</code>	322–327
	60, 62, 63, 68–72, 74, 105,	<code>\glxtrLatinS</code>	322–327
	175, 176, 185, 188–194, 207, 230, 244,	<code>\glxtrLatinT</code>	322–327
	265, 266, 269, 271, 273, 283, 284, 288, 290	<code>\glxtrLatinThorn</code>	326
<code>\glxtrinclinkcounter</code>	141	<code>\glxtrLatinX</code>	322–327
<code>\glxtrindexaliased</code>	75, 76	<code>\glxtrlocationhyperlink</code>	117
<code>\glxtrindexseealso</code>	42, 43	<code>\glxtrlocrangefmt</code>	117
<code>\glxtrinithyperoutside</code>	58	<code>\GLSxtrlong</code>	17, 18, 302, 303
<code>\glxtrinitwrgloss</code>	58, 136	<code>\Glsxtrlong</code>	17, 18, 303
<code>\glxtrinitwrglossbeforefalse</code>	57	<code>\glxtrlong</code>	17, 18, 302
<code>\glxtrinitwrglossbeforetrue</code>	57	<code>\glxtrlonghyphen</code>	283, 284
<code>\Glsxtrinilinefullformat</code>	183, 185, 198,	<code>\glxtrlonghyphennoshort</code>	280, 281
	199, 206, 208, 209, 211, 213, 215, 221,	<code>\glxtrlonghyphenshort</code>	278
	223–225, 227, 229, 231, 236–239, 241,	<code>\glxtrlongnoshortdescname</code>	214, 261, 279
	243, 245, 253, 255–257, 259, 260, 262,	<code>\glxtrlongnoshortname</code>	
	265, 266, 270, 272, 280, 284, 289, 292, 311		216, 224, 239, 256, 258, 281
<code>\glxtrinilinefullformat</code>		<code>\GLSxtrlongpl</code>	17, 18, 303, 304
	183, 185, 186, 198, 199,	<code>\Glsxtrlongpl</code>	17, 18, 304
	206, 208, 209, 211, 213, 215, 221, 223–	<code>\glxtrlongpl</code>	17, 18, 303
	225, 227, 229, 231, 235, 237–239, 241,	<code>\glxtrlongshortdescname</code>	
	243, 245, 253–255, 257, 258, 260, 262,		202, 218, 233, 247, 249, 278, 284
	264, 266, 270, 272, 280, 284, 289, 291, 311	<code>\glxtrlongshortdescsort</code>	
<code>\Glsxtrinilinefullplformat</code>	184, 187, 198,		202, 218, 233, 247, 249, 274, 278, 284
	199, 207, 208, 210, 212, 213, 215, 222–	<code>\glxtrlongshortname</code>	
	225, 227, 229, 231, 236–238, 240, 241,		200, 217, 232, 246, 247, 268, 269, 277, 282
	243, 245, 253, 255–257, 259, 261, 262,	<code>\glxtrlongshortuserdescname</code>	271, 274
	265, 266, 270, 273, 280, 284, 289, 292, 312	<code>\glxtrmarkhook</code>	293, 294
<code>\glxtrinilinefullplformat</code>		<code>\glxtrMathItalicAlpha</code>	331, 335, 336
	183, 184, 186, 187, 198, 199,	<code>\glxtrMathItalicBeta</code>	331, 335, 336
	206, 208, 209, 211, 213, 215, 221, 223–	<code>\glxtrMathItalicChi</code>	331, 332, 336, 337
	225, 227, 229, 231, 236–239, 241, 243,	<code>\glxtrMathItalicDelta</code>	331, 335, 336
	245, 253, 254, 256, 257, 259, 260, 262,	<code>\glxtrMathItalicEpsilon</code>	331, 332, 335, 336
	264, 266, 270, 272, 280, 284, 289, 292, 311	<code>\glxtrMathItalicEta</code>	331, 332, 335, 336
<code>\glxtrininsertinsidefalse</code>	200	<code>\glxtrMathItalicGamma</code>	331, 335, 336

<code>\glxtrMathItalicIota</code> ..	331, 332, 335, 336	<code>\glxtrpostdescription</code>	
<code>\glxtrMathItalicKappa</code> .	331, 332, 335, 336	110, 111, 157, 173, 359
<code>\glxtrMathItalicLambda</code>	331, 332, 335, 336	<code>\glxtrposthyphenlong</code>	288, 290
<code>\glxtrMathItalicMu</code>	331, 332, 335, 336	<code>\glxtrposthyphenshort</code>	283, 284
<code>\glxtrMathItalicNu</code>	331, 332, 335, 336	<code>\glxtrposthyphensubsequent</code>	
<code>\glxtrMathItalicOmega</code> .	331, 332, 336, 337	283, 284, 288, 290
<code>\glxtrMathItalicOmicron</code>	331, 332, 335, 336	<code>\glxtrpostlink</code>	174
<code>\glxtrMathItalicPhi</code> ...	331, 332, 335, 337	<code>\glxtrpostlinkendsentence</code>	174
<code>\glxtrMathItalicPi</code>	331, 332, 335, 336	<code>\glxtrpostlinkhook</code>	174
<code>\glxtrMathItalicPsi</code> ...	331, 332, 336, 337	<code>\glxtrpostlocalreset</code>	90, 92, 100
<code>\glxtrMathItalicRho</code> ...	331, 332, 335, 336	<code>\glxtrpostlocalunset</code>	89, 91, 99, 100
<code>\glxtrMathItalicSigma</code> .	331, 332, 335, 336	<code>\glxtrpostlongdescription</code>	34
<code>\glxtrMathItalicTau</code> ...	331, 332, 335, 336	<code>\glxtrpostnamehook</code>	161–164, 166
<code>\glxtrMathItalicTheta</code> .	331, 332, 335, 336	<code>\GlsXtrPostNewAbbreviation</code>	
<code>\glxtrMathItalicUpsilon</code>	331, 332, 335, 337	182, 198, 199, 201–205,
<code>\glxtrMathItalicXi</code>	331, 332, 335, 336		207, 209–214, 216, 217, 219–222, 224,
<code>\glxtrMathItalicZeta</code> ..	331, 332, 335, 336		228, 229, 232–235, 237, 239, 242, 244,
<code>\glxtrnewabbrevpresetkeyhook</code>	181		246–249, 251–254, 256, 258, 259, 261,
<code>\glxtrnewnumber</code>	18		263, 265, 268, 269, 271, 273, 274, 276,
<code>\glxtrnewsymbol</code>	18		278, 279, 281, 282, 284, 286–288, 290–292
<code>\glxtrNoGlossaryWarning</code>	20, 118	<code>\glxtrpostreset</code>	90, 92, 100
<code>\GlsXtrNoGlsWarningAutoMake</code>	122	<code>\glxtrpostunset</code>	89, 91, 99
<code>\GlsXtrNoGlsWarningBuildInfo</code>	122	<code>\glxtrprelocation</code>	
<code>\GlsXtrNoGlsWarningCheckFile</code>	122	349, 351, 353, 355, 357, 359, 382
<code>\GlsXtrNoGlsWarningEmptyMain</code>	122	<code>\glxtrprotectlinks</code>	79, 80
<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	122	<code>\GlsXtrRecordCounter</code>	11
<code>\GlsXtrNoGlsWarningEmptyStart</code>	121	<code>\glxtrrecordtriggervalue</code>	136
<code>\GlsXtrNoGlsWarningHead</code>	121	<code>\glxtrregularfont</code>	54, 61
<code>\GlsXtrNoGlsWarningMisMatch</code>	122	<code>\glxtrresourcecount</code>	123
<code>\GlsXtrNoGlsWarningNoOut</code>	122	<code>\glxtrresourcefile</code>	123
<code>\GlsXtrNoGlsWarningTail</code>	122	<code>\glxtrresourceinit</code>	123
<code>\glxtrnopostpunc</code>	110	<code>\glxtrrestoremarkhook</code>	293, 294
<code>\glxtronlydescname</code>	292	<code>\glxtrrestorepostpunc</code>	110, 111
<code>\glxtronlydescsort</code>	292	<code>\glxtrscfont</code>	217
<code>\glxtronlyname</code>	291	<code>\glxtrscsuffix</code>	
<code>\glxtronlysuffix</code>	291	218, 219, 221, 222, 225, 226, 228, 230
<code>\glxtrorg@ifKV@glslink@hyper</code>	55	<code>\GlsXtrSetActualChar</code>	170
<code>\glxtrorglong</code>	180, 202, 279	<code>\glxtrsetaliasnoindex</code>	13, 14, 76
<code>\glxtrorgshort</code>	180, 202	<code>\GlsXtrSetEncapChar</code>	170
<code>\GLSxtrp</code>	85	<code>\GlsXtrSetEscChar</code>	170
<code>\Glsxtrp</code>	85	<code>\glxtrsetfieldifexists</code>	31, 32
<code>\glxtrp</code>	85, 87	<code>\GlsXtrSetLevelChar</code>	170
<code>\glxtrparen</code>	175, 183,	<code>\glxtrsetpopts</code>	84
	201–204, 206–213, 215, 217–225, 227,	<code>\glxtrsetupfulldefs</code>	
	229, 231–241, 243, 245–257, 259–262,	185–187, 207, 230, 244, 266
	264–267, 277, 278, 280, 282, 285–287, 292	<code>\GLSxtrshort</code>	17, 18, 88, 89, 297
<code>\Glsxtrpl</code>	49	<code>\Glsxtrshort</code>	17, 18, 297, 298
<code>\glxtrpl</code>	49	<code>\glxtrshort</code>	17, 18, 297
		<code>\glxtrshortdescname</code> ...	211, 222, 236, 254

<code>\glxtrshorthyphen</code>	289	<code>\glxtrrtitleshort</code>	295, 296, 306
<code>\glxtrshorthyphenlong</code>	286	<code>\Glsxtrrtitleshortpl</code>	295, 296, 307
<code>\glxtrshortlongdescname</code>		<code>\glxtrrtitleshortpl</code>	295, 296, 306
..... 204, 220, 234, 250, 252, 286, 289		<code>\Glsxtrrtitletext</code>	295, 296, 308
<code>\glxtrshortlongdescsort</code>		<code>\glxtrrtitletext</code>	295, 296, 307, 308
.... 204, 220, 234, 250, 252, 275, 286, 289		<code>\GlsXtrTotalRecordCount</code>	136
<code>\glxtrshortlongname</code>		<code>\glxtrtreetopindent</code>	364, 373
203, 219, 233, 249, 251, 271, 274, 285, 288		<code>\glxtrundefaction</code> ..	7, 13, 26, 34, 35, 37, 38
<code>\glxtrshortlonguserdescname</code> ..	273, 275	<code>\glxtrundeftag</code>	25, 113
<code>\glxtrshortnolongname</code> .	209, 221, 235, 253	<code>\glxtrunsrtdo</code>	130
<code>\GlsXtrshortpl</code>	17, 18, 297, 298	<code>\glxtrUpAlpha</code>	330, 335, 336
<code>\Glsxtrshortpl</code>	17, 18, 298	<code>\glxtrUpBeta</code>	330, 335, 336
<code>\glxtrshortpl</code>	17, 18, 297	<code>\glxtrUpChi</code>	330, 331, 336, 337
<code>\glxtrsmfont</code>	231	<code>\glxtrUpDelta</code>	330, 335, 336
<code>\glxtrsmsuffix</code>		<code>\glxtrUpDigamma</code>	330, 331, 335
.... 232, 234, 235, 237, 239, 240, 242, 244		<code>\glxtrUpEpsilon</code>	330, 335, 336
<code>\Glsxtrsubsequentfmt</code>		<code>\glxtrUpEta</code>	330, 335, 336
..... 195, 198, 214, 225, 226,		<code>\glxtrUpGamma</code>	330, 335, 336
239, 241, 257, 258, 260, 262, 280, 283, 288		<code>\glxtrUpIota</code>	330, 335, 336
<code>\glxtrsubsequentfmt</code>		<code>\glxtrUpKappa</code>	330, 335, 336
..... 195, 198, 214, 225, 226,		<code>\glxtrUpLambda</code>	330, 335, 336
239, 240, 256, 258, 260, 262, 280, 283, 288		<code>\glxtrUpMu</code>	330, 335, 336
<code>\Glsxtrsubsequentplfmt</code>		<code>\glxtrUpNu</code>	330, 335, 336
..... 195, 198, 214, 225, 226,		<code>\glxtrUpOmega</code>	330, 331, 336, 337
239, 241, 257, 258, 260, 262, 280, 283, 288		<code>\glxtrUpOmicron</code>	330, 331, 335, 336
<code>\glxtrsubsequentplfmt</code>		<code>\glxtrUpPhi</code>	330, 331, 336, 337
..... 195, 198, 214, 225, 226,		<code>\glxtrUpPi</code>	330, 331, 335, 336
239, 241, 256, 258, 260, 262, 280, 283, 288		<code>\glxtrUpPsi</code>	330, 331, 336, 337
<code>\glxtrsuppllocationurl</code>	118	<code>\glxtrUpRho</code>	330, 331, 335, 336
<code>\glxtrtagfont</code>	173	<code>\glxtrUpSigma</code>	330, 331, 335, 336
<code>\Glsxtrrtitlefirst</code>	295, 296, 309	<code>\glxtrUpTau</code>	330, 331, 335, 336
<code>\glxtrrtitlefirst</code>	295, 296, 309	<code>\glxtrUpTheta</code>	330, 335, 336
<code>\Glsxtrrtitlefirstplural</code> ..	295, 296, 309, 310	<code>\glxtrUpUpsilon</code>	330, 331, 335, 337
<code>\glxtrrtitlefirstplural</code>	295, 296, 309	<code>\glxtrUpXi</code>	330, 331, 335, 336
<code>\Glsxtrrtitlefull</code>	295, 296, 311	<code>\glxtrUpZeta</code>	330, 335, 336
<code>\glxtrrtitlefull</code>	295, 296, 311	<code>\GlsXtrUseAbbrStyleFmts</code>	
<code>\Glsxtrrtitlefullpl</code>	295, 296, 312 202, 204, 211–213,	
<code>\glxtrrtitlefullpl</code>	295, 296, 311, 312	216, 217, 219, 221, 224, 233, 235, 238,	
<code>\Glsxtrrtitlelong</code>	295, 296, 310	247, 249, 251, 252, 255, 260, 263, 271,	
<code>\glxtrrtitlelong</code>	295, 296, 310	273, 274, 276, 279–281, 285, 287, 290, 293	
<code>\Glsxtrrtitlelongpl</code>	295, 296, 311	<code>\GlsXtrUseAbbrStyleSetup</code>	
<code>\glxtrrtitlelongpl</code>	295, 296, 310 210, 212, 213, 215,	
<code>\Glsxtrrtitlename</code>	295, 296, 307	216, 224, 226, 238, 240, 255, 259, 260, 263	
<code>\glxtrrtitlename</code>	295, 296, 307	<code>\glxtruserfield</code>	267
<code>\glxtrrtitleorpdforheading</code>		<code>\glxtruserparen</code>	268–276
..... 23, 126, 127, 295, 296		<code>\glxtrusersuffix</code>	268, 269, 272, 275
<code>\Glsxtrrtitleplural</code>	295, 296, 308	<code>\glxtruseseealsoformat</code>	40, 41
<code>\glxtrrtitleplural</code>	295, 296, 308	<code>\glxtruseseeformat</code>	39
<code>\Glsxtrrtitleshort</code>	295, 296, 306	<code>\GlsXtrWarnDeprecatedAbbrStyle</code> ..	178, 199

<code>\GlsXtrWarning</code>	47–49	<code>\ifglstryentryexists</code>	8, 37, 38, 47–49, 51, 52, 60, 132, 154, 173, 174
<code>\glstrword</code>	180	<code>\ifglstrfieldeq</code>	153
<code>\glstrwordsep</code> ..	180, 277, 279, 282, 285, 287	<code>\ifglstrhasfield</code>	28, 267
<code>\glstrwrglossmark</code>	22	<code>\ifglstrhaslong</code>	96, 139, 140
H			
<code>\hangindent</code> .	361, 362, 364, 373–375, 379, 380	<code>\ifglstrhasparent</code>	126, 127, 129, 132, 366, 368, 369
<code>\hbox</code>	348	<code>\ifglstrhasshort</code>	40, 54, 60
<code>\hfill</code>	348	<code>\ifglstrhasymbol</code>	175, 360–362, 364
<code>\href</code>	79	<code>\ifglstrindexonlyfirst</code>	76
<code>\hsize</code>	50, 51	<code>\ifglstrnogroupskip</code> ..	349, 351–361, 363, 375, 383
<code>\hss</code>	348	<code>\ifglstrnonumberlist</code>	54
<code>\hyperlink</code>	79, 80	<code>\ifglstrnopostdot</code>	15, 110
<code>\hyperpage</code>	166	<code>\ifglstrsanitizesort</code>	108
<code>\hyperref</code>	79, 118	<code>\ifglstrsubentrycounter</code>	32
<code>hyperref</code> package	80, 166, 293, 306	<code>\ifglstrused</code>	44, 74, 76, 77, 92, 101, 105, 194, 366–368, 370–372
I			
<code>\if</code>	47	<code>\ifglstrxindy</code>	118, 120
<code>\if@glstr@autoseeindex</code>	23, 24, 39, 42	<code>\ifglstr@hyperoutside</code>	59
<code>\if@glstr@format@override</code>	167	<code>\ifglstrtrinitwrglossbefore</code> ...	57, 59, 137
<code>\if@glstr@docdefrestricted</code>	45	<code>\ifglstrinsertinside</code> ..	188–194, 196, 197, 201, 203, 204, 206–215, 218, 220– 248, 250–266, 268–270, 272, 273, 275, 277, 279, 282, 284, 285, 287, 289, 291, 292
<code>\if@glstr@indexcrossrefs</code>	14, 44	<code>\ifHy@hyperindex</code>	166
<code>\ifblank</code>	26, 47–49, 105	<code>\ifinlistcs</code>	29, 45
<code>\ifcase</code> ..	7, 12, 19, 20, 22, 46, 57, 111, 360, 385	<code>\ifinner</code>	23
<code>\ifcsdef</code> ..	25, 32–37, 59, 72, 73, 84–89, 97, 109, 114, 115, 127, 131, 133, 135, 141, 159, 160, 162–165, 175, 178, 194, 198, 351–358	<code>\ifKV@glslink@hyper</code>	55, 58, 59
<code>\ifcsstring</code>	25, 154, 197	<code>\ifKV@glslink@local</code>	56, 137
<code>\ifcsundef</code>	30, 32–37, 45, 50, 52, 80, 91, 97–100, 114, 115, 118, 130, 141, 142, 154, 197–200, 348, 365, 366, 373, 387	<code>\ifKV@glslink@noindex</code>	8, 9, 11, 28, 76
<code>\ifcsvoid</code>	43, 153	<code>\ifmmode</code>	23
<code>\ifdef</code> ..	13, 18, 24, 28, 38, 40, 41, 50, 51, 75, 78, 79, 86–88, 109, 112, 113, 124, 133, 156, 157, 170, 173, 267, 294, 306–312, 345, 348–351, 359–363, 375–380, 383, 386, 387	<code>\ifnum</code> .	14, 93, 101, 114, 123, 136, 361, 362, 374
<code>\ifdefempty</code>	7–9, 30, 34–36, 39, 40, 58, 60, 91, 102, 103, 106, 109, 117, 129, 131, 136, 172, 179, 194, 384	<code>\ifstrempy</code>	127, 133, 142
<code>\ifdefequal</code>	45, 122, 131, 164	<code>\ifstrequal</code>	16, 21
<code>\ifdefstring</code>	6, 32, 167, 172, 383	<code>\ifthenelse</code>	122
<code>\ifdefvoid</code> ...	39, 42–44, 79, 96, 114, 118, 132	<code>\IfTrackedLanguageFileExists</code>	312
<code>\ifdim</code>	50, 51, 105, 365–373	<code>\ifundef</code>	30, 106, 172, 173, 345
<code>\IfFileExists</code> ..	21, 118, 122, 123, 125, 345, 347	<code>\ifx</code> ..	8–10, 41, 50, 51, 105, 106, 109, 116, 117, 125, 167, 168, 170, 177, 179–181, 276, 381
<code>\ifglossaryexists</code>	38	<code>\immediate</code>	92, 101, 118, 119, 125
<code>\ifglssacronym</code>	17, 122	<code>\index</code>	167
<code>\ifglssacrshortcuts</code>	19	<code>\indexspace</code> .	349, 360–363, 375–381, 383, 386
<code>\ifglssautomake</code>	109, 122, 125	<code>\input</code>	312
<code>\ifglstryentrycounter</code>	32	<code>\inputencodingname</code>	124
		<code>\istfilename</code>	106
		<code>\item</code>	120, 121, 348–351, 359–361, 376, 377
J			
		<code>\jobname</code>	118, 120–123, 125

K	
<code>\key@ifundefined</code> .	11, 12, 26, 27, 72, 128, 131
<code>\KV@glslink@hyperfalse</code> ..	61, 74, 75, 80, 81
<code>\KV@glslink@hypertrue</code>	80
<code>\KV@glslink@noindexfalse</code>	75
<code>\KV@glslink@noindextrue</code>	75, 81
L	
<code>\L</code>	327, 330
<code>\l</code>	330
<code>\LaTeX</code>	120, 121
<code>\leaders</code>	348
<code>\leavevmode</code>	34, 58
<code>\let</code>	5, 7–13, 15, 17–19, 23–25, 28, 30, 33, 34, 45, 46, 50, 52, 53, 55, 56, 58–78, 80, 81, 84, 90–92, 99–115, 123, 125, 128, 129, 132, 136, 137, 142, 159–169, 172, 173, 177–181, 185–194, 196–198, 207, 230, 244, 266, 293–296, 345, 359, 360, 364, 366, 377, 384–387
<code>\letabbreviationstyle</code> ..	207, 209, 210, 212, 215, 216, 222, 223, 236, 238, 254, 255
<code>\letcs</code>	26, 30, 39, 40, 44, 59, 72, 112–114, 131, 132, 159–166, 387
<code>\levelchar</code>	170
<code>\listadd</code>	97
<code>\listbreak</code>	172
<code>\listcsadd</code>	29
<code>\listcseadd</code>	29, 98
<code>\listcsgadd</code>	29, 45
<code>\listcsxadd</code>	29, 97
<code>\loadglsentries</code>	46, 120
<code>\long</code>	33, 34
M	
<code>\MakeAcronymsAbbreviations</code>	104
<code>\makeatletter</code>	118, 123, 169
<code>\makeatother</code>	169
<code>\makebox</code>	348, 374, 375
<code>\makefirstuc</code>	173
<code>makeglossaries</code>	112
<code>\makeglossaries</code>	105, 119–122, 125
<code>\makeglossary</code>	106
<code>makeindex</code>	388
<code>makeindex</code>	13, 105
<code>\makenoidxglossaries</code>	120
<code>\MakeTextUppercase</code>	295
<code>\MakeUppercase</code>	295, 296
<code>\marginpar</code>	23
<code>\markboth</code>	294
<code>\markright</code>	294
<code>\mathit</code>	314
<code>\mathrm</code>	313–315
<code>\maxdimen</code>	50, 51
<code>\mbox</code>	350, 351, 374
<code>\medskip</code>	122, 130
<code>\MessageBreak</code>
<code>\mfirstuc package</code>	172
<code>\mfirstucMakeUppercase</code>
<code>\mfu@checkword@arg</code>	172, 173
<code>\mfu@checkword@do</code>	173
N	
<code>\NeedsTeXFormat</code>	5, 313, 347, 382
<code>\new@glossaryentry</code>	46, 108
<code>\new@ifnextchar</code>	27, 73, 74, 95, 133, 134, 137–139, 176, 184–193
<code>\newabbr</code>	17, 18
<code>\newabbreviation</code>	17, 18
<code>\newabbreviationhook</code>	182
<code>\newabbreviationstyle</code>	200, 202–205, 207, 209–222, 224, 226, 228, 229, 232–236, 238, 240, 242, 243, 246–256, 258–261, 263, 265, 268, 269, 271, 273–275, 277–279, 281, 282, 284–286, 288, 289, 291, 292
<code>\newacronym</code>	103, 104
<code>\newacronymhook</code>	103
<code>\newacronymstyle</code>	104
<code>\newcommand</code>	5–8, 10–12, 14–40, 42, 44–50, 52–54, 57, 58, 60, 61, 72–81, 84, 86–91, 93, 95–99, 101, 102, 104, 105, 109–121, 123–158, 164–180, 182–194, 196–200, 202, 204, 205, 209, 211, 214, 216, 217, 231, 245, 246, 267, 268, 270, 273, 277, 279, 281, 282, 285, 287, 290, 292, 294–313, 316–344, 347, 349, 359, 363–366, 373, 374, 382, 383, 386, 387
<code>\newcount</code>	14, 123, 133
<code>\newcounter</code>	141
<code>\newentry</code>	18
<code>\newglossary</code>	16, 106
<code>\newglossaryentry</code>
<code>\newglossaryentry options</code>	18, 46, 91, 99, 103, 156, 157, 182
<code>alias</code>	15, 39, 41–44

desc	146, 147, 151	\ns@GLSxtrlongpl	193
descplural	147, 151, 152	\ns@Glsxtrlongpl	193
first	78, 144, 150, 200, 301, 302, 308, 388	\ns@glxtrlongpl	192
firstplural		\ns@GLSxtrshort	189
	144, 145, 150, 151, 200, 301, 302, 309, 388	\ns@Glsxtrshort	188
group	131, 132	\ns@glxtrshort	187, 188
loclist	29	\ns@GLSxtrshortpl	192
long	149, 152, 310	\ns@Glsxtrshortpl	191
longplural	149, 152, 310	\ns@glxtrshortpl	191
name 40, 142, 143, 149, 150, 167, 298, 299, 307		\null	20
plural	143, 144, 150, 200, 300, 308	\number	98–100, 123, 133
see	15, 24, 39, 41, 44, 46, 106	\numexpr	98, 100
seealso	15, 39–41, 43, 44, 399		
short	148, 152, 179	O	
shortplural	148, 152, 179	\O	327, 329
symbol	145, 151	\o	329
symbolplural	146, 151	\or	7, 13, 19, 20, 22, 46, 57, 360, 385
text	78, 143, 150, 200, 202, 299, 300, 307	\org@glossaryentrynumbers	51, 110
\newglossarystyle	384	\org@glossarytitle	109
\newif	57, 166, 200	\org@ifKV@glslink@hyper	58, 59
\newlength	364		
\newnum	18	P	
\newrobustcmd	27, 28, 30–32, 40,	\p@glsl@hyp@opt	77
	41, 73, 74, 84–86, 95, 110, 114, 126, 127,	package options:	
	130, 131, 133–135, 137–139, 165, 172,	abbreviations	16, 17
	173, 184–194, 276, 294, 297–305, 366–372	accsupp	20, 142
\newsym	18	acronym	17
\newterm	156	automake	109, 120, 125
\newtoks	178, 179	true	125
\newwrite	106	autoseeindex	24
\nobreak	350, 351, 360–363, 376–379	false	23
\NoCaseChange	86–89, 127, 296–305	debug	
\noexpand	10,	showtargets	79
	11, 21, 40, 42, 43, 103, 118, 119, 123,	docdef	14, 45, 46, 91, 99
	129, 130, 132, 141, 168, 169, 182, 347, 385	false	46
\nofiles	121	restricted	14
\noindent	122, 362, 363, 377–380	true	46
\nopagebreak	360–363, 375–382, 386	docdefs	
\nopostdesc	34, 48, 49, 110, 157	restricted	45
\nr	7, 12, 14, 19, 20, 22, 57, 111	nonumberlist	51
\ns@GLSxtrfull	186	nopostdot	15
\ns@Glsxtrfull	185	false	15
\ns@glxtrfull	184	numbers	18
\ns@GLSxtrfullpl	187	postdot	15
\ns@Glsxtrfullpl	186	record	7, 12, 45, 54, 105, 122, 123, 397
\ns@glxtrfullpl	186	alsoindex	8, 10
\ns@GLSxtrlong	190	only	7, 8
\ns@Glsxtrlong	190	shortcuts	19
\ns@glxtrlong	189	ac	19
		all	19

\rGlspl	135	\TeX	120
\rglspl	135	\texorpdfstring	28, 86–88, 294, 306–312
\rGLSplformat	139	textcase package	293
\rGlsplformat	138	\textsc	217
\rglsplformat	138, 140	\textsmaller	231
\romannumeral	364–366, 373	\texttt	23, 119–122
S			
\s@glsxtrfmt	27	\the	103,
\s@gls@hyp@opt	77	117, 123, 170, 171, 182, 200–205, 207,	
\s@glsxtr@enabletagging	171, 172	209–214, 216–222, 224, 228–230, 232–	
\s@glsxtrfmt	27	235, 237, 239, 242–244, 246–254, 256,	
\s@glsxtrifhasfield	30	258, 259, 261, 263–265, 268–279, 281–292	
\s@printunsrtglossary	128, 130	\theglentrycounter	8–10, 58, 60, 137
\seealso	40, 41	\theHglentrycounter	8–10, 58, 60, 137
\seename	39	\theindex	166
\setabbreviationstyle	104, 202, 210	\this@dialect	312, 313, 345
\setacronymstyle	104, 105	\thisgrptitle	386
\setentrycounter	116	\toks@	117, 170, 171
\SetGenericNewAcronym	104	tracklang package	124, 316
\setglossarystyle	22,	\TrackLangGetDefaultScript	345
109, 348–351, 360, 362, 363, 375–381, 384		\TrackLangIfHasDefaultScript	345
\setkeys	9,	\TrackLangRequireDialectPrefix	345
21, 24, 28, 58, 60, 76, 103, 109, 136, 180, 181		U	
\setlength	50, 51,	\u	313
359, 361, 362, 364, 374, 375, 377–379, 384		\undef	13, 14, 172
\settowidth	105, 364–373	\underline	173
\setupglossaries	5, 24	\unskip	34, 45, 348
\sfcode	15, 16, 175, 359	upgreek package	314
\small	23	\usepackage	121, 122
\space	6, 11, 41, 46,	V	
47, 49, 76, 90–93, 99, 101, 102, 104–109,		\val	7, 12, 14, 19, 20, 22, 57, 111
119, 121, 125, 129, 131, 133, 175, 179,		W	
183, 202, 348, 349, 359–362, 364, 373, 382		\warn@nomakeglossaries	107
\spacefactor	15, 16, 175, 181, 359	\warn@noprintglossary	107, 110
\stepcounter	141	\write	41, 92, 101, 106, 118, 119, 125
\string	6, 10, 11, 41,	X	
45–47, 49, 53, 59, 73, 76, 85, 90–93, 99,		\x	117, 141
101, 102, 104–109, 118–126, 129, 131,		\xcapitalisewords	158
133, 160, 162–165, 167, 313, 316–345, 387		\xdef	110, 130
\strut	348–358, 362	\xifinlist	97
\subglossentry		\xifinlistcs	29
.... 110, 132, 348–358, 360–362, 374, 385		xindy	388
\subitem	359, 360	xindy	13, 105
\subsubitem	360	xkeyval package	5
T		\XKV@checkchoice	53
\tablehead	355–358	\XKV@plfalse	53
\tabletail	355–358	\XKV@resa	53, 54
\tabularnewline	351–359	\XKV@sttrue	53