

glossaries-extra.sty v1.29: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2018-04-09

Abstract

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

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

Contents

1	Main Package Code (glossaries-extra.sty)	5
1.1	Package Initialisation and Options	5
1.2	Extra Utilities	27
1.3	Modifications to Commands Provided by glossaries	34
1.3.1	Existence Checks	39
1.3.2	Document Definitions	46
1.3.3	Existing Glossary Style Modifications	52
1.3.4	Entry Formatting, Hyperlinks and Indexing	56
1.3.5	Entry Counting	91
1.3.6	Acronym Modifications	104
1.3.7	Indexing and Displaying Glossaries	107
1.4	Link Counting	143
1.5	Integration with glossaries-accsupp	144
1.6	Categories	155
1.7	Abbreviations	180
1.7.1	Abbreviation Styles Setup	199
1.7.2	Predefined Styles (Default Font)	202
1.7.3	Predefined Styles (Small Capitals)	219
1.7.4	Predefined Styles (Fake Small Capitals)	234
1.7.5	Predefined Styles (Emphasized)	248
1.7.6	Predefined Styles (User Parentheses Hook)	269
1.7.7	Predefined Styles (Hyphen)	278
1.7.8	Predefined Styles (No Short on First Use)	292
1.8	Using Entries in Headings	295
1.9	Multi-Lingual Support	314
1.10	glossaries-extra-bib2gls.sty	315
2	Style Adjustments (glossaries-extra-stylemods.sty)	351
2.1	Package Initialisation	351
2.2	List-Like Styles	352
2.3	Longtable Styles	355
2.4	Long Ragged Styles	357
2.5	Supertabular Styles	359
2.6	Super Ragged Styles	361
2.7	Inline Style	363
2.8	Tree Styles	363
2.9	Multicolumn Styles	380

3 bookindex style (glossary-bookindex.sty)	386
3.1 Package Initialisation and Options	386
Glossary	392
Change History	393
Index	411

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2018/04/09 v1.29 (NLCT)]
```

Requires xkeyval to define package options.

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

Requires etoolbox package.

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

Has glossaries already been loaded?

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

```
6 {%
```

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

```
7 \newcommand{\glxstr@doption}[1]{\setupglossaries{#1}}%
```

```
8 \let\@glxstr@declareoption\@gls@declareoption
```

```
9 }
```

```
10 {%
```

Not already loaded, so pass options to glossaries.

```
11 \newcommand{\glxstr@doption}[1]{%
```

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

```
13 }%
```

Set the defaults.

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

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

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

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

```
18 {}%
```

```
19 {%
```

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

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

```
22 {}%
```

```
23 }%
```

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

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

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

```
27 }
```

```
28 }
```

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

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

```

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

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

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

`\glstr@recordsee` Does nothing by default.

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

`\ultnumberformat`

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

`\ultNumberFormat`

```

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

```

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

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

```

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

```

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

```

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

```

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

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

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

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

Increment associated counter.

```
124   \glsxtr@inc@wrglossaryctr{#2}%  
125   \begingroup
```

Save the label in case it's needed.

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

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

```
132     \let\@gls@counter\glscounter
```

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

```
133     \@gls@setdefault@glslink@opts  
134     \setkeys{#3}{#1}%  
135     \ifKV@glslink@noindex  
136     \else  
137       \glswriteentry{#2}%  
138     {%
```

Check if thevalue has been set.

```
139       \ifdefempty{\@glsxtr@thevalue}%  
140       {%
```

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

```
141         \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue  
142         \else  
143           \let\theHglentrycounter\@glsxtr@theHvalue  
144         \fi
```

Save the entry counter.

```
145         \glsxtr@saveentrycounter
```

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

```
146         \let\@@do@@wrglossary\@glsxtr@dorecord  
147         }%  
148       {%
```

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.)

```
149         \let\theHglentrycounter\@glsxtr@thevalue  
150         \let\theHglentrycounter\@glsxtr@theHvalue
```

```

151         \let\@do@wrglossary\@glxtr@dorecordnodefer
152     }%
153     \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
154         \glxtr@do@wrglossary{#2}%
155     \else
        No need to escape special characters.
156         \@do@wrglossary
157     \fi
158 }%
159 \fi
160 \endgroup
161 }%
162 }

```

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

```

163 \newcommand*\@glxtr@dorecord{%
164     \global\let\@glsrecordlocref\theglentrycounter
165     \let\@glxtr@orgprefix\@glo@counterprefix
166     \ifx\theglentrycounter\theHglentrycounter
167         \def\@glo@counterprefix{}%
168     \else
169         \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
170             {\theglentrycounter}{\theHglentrycounter}}%
171     }%
172     \do@gls@getcounterprefix
173 \fi
174 \ifdefstring\@gls@counter{page}%
175 {%

```

Protect the location from premature expansion.

```

176     \protected@write\@auxout{\let\@glsrecordlocref\relax}{\string\glxtr@record
177         {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}}%
178         {\@glsrecordlocref}}%
179 }%
180 {%

```

Don't protect the location from premature expansion.

```

181     \protected@write\@auxout{}{\string\glxtr@record
182         {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}}%
183         {\@glsrecordlocref}}%
184 }%
185 \@glxtr@counterrecordhook
186 \let\@glo@counterprefix\@glxtr@orgprefix
187 }

```

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

```

188 \newcommand*\@glxtr@dorecordnodefer{%
189   \ifx\theglsentrycounter\theHglentrycounter
190     \protected@write\@auxout{}\string\glxtr@record
191       {\@gls@label}{\@gls@counter}{\@glsnumberformat}%
192       {\theglsentrycounter}}%
193   \else
194     \edef\@do@gl@getcounterprefix{\noexpand\@gl@getcounterprefix
195       {\theglsentrycounter}{\theHglentrycounter}}%
196     }%
197     \@do@gl@getcounterprefix
198     \protected@write\@auxout{}\string\glxtr@record
199       {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
200       {\theglsentrycounter}}%
201   \fi
202   \@glxtr@counterrecordhook
203 }

```

r@recordcounter

```

204 \newcommand*\@@glxtr@recordcounter{%
205   \@glxtr@noop@recordcounter
206 }

```

p@recordcounter

```

207 \newcommand*\@glxtr@noop@recordcounter}[1]{%
208   \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
209     requires record=only or record=alsoindex package option}{}%
210 }

```

p@recordcounter

```

211 \newcommand*\@glxtr@op@recordcounter}[1]{%
212   \eappto\@glxtr@counterrecordhook{\noexpand\@glxtr@docounterrecord{#1}}%
213 }

```

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

```

214 \newcommand*\@glxtr@recordsee}[2]{%
215   \@glxtr@wrglossmark
216   \def\@gl@xref{#2}%
217   \@onelevel@sanitize\@gl@xref
218   \protected@write\@auxout{}\string\glxtr@recordsee{#1}{\@gl@xref}}%
219 }

```

srtglossaryunit

```

220 \newcommand{\printunsrtglossaryunit}{%
221   \print@noop@unsrtglossaryunit
222 }

```

tr@setup@record Initialise.

```

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

```

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

```
224 \newcommand*{\glstr@indexonly@saveentrycounter}{%
225   \ifKV@glslink@noindex
226   \else
227     \glstr@saveentrycounter
228   \fi
229 }
```

`@addloclistfield`

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

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

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

Add a key to store the group heading.

```
253 \key@ifundefined{glossentry}{group}%
254 {%
255   \define@key{glossentry}{group}{\def\@glo@group{##1}}%
256   \appto\@gls@keymap{,{group}{group}}%
257   \appto\@newglossaryentryprehook{\def\@glo@group{}}%
258   \appto\@newglossaryentryposthook{%
259     \gls@assign@field{\@glo@label}{group}{\@glo@group}%
260   }%
261   \glssetnoexpandfield{group}%
262 }%
263 }%
264 }
```

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

```

265 \newcommand*{\@glxtr@record@setting}{off}

tting@alsoindex
266 \newcommand*{\@glxtr@record@setting@alsoindex}{alsoindex}

rd@setting@only
267 \newcommand*{\@glxtr@record@setting@only}{only}

ord@setting@off
268 \newcommand*{\@glxtr@record@setting@off}{off}

Now define the record package option.
269 \define@choicekey{glossaries-extra.sty}{record}
270 [\@glxtr@record@setting\glxtr@record@nr]%
271 {off,only,alsoindex}%
272 [only]%
273 {%
274 \ifcase\glxtr@record@nr\relax

Don't record.
275 \def\glxtr@setup@record{%
276 \renewcommand*{\@do@seeglossary}{\@glxtr@doseeglossary}%
277 \renewcommand*{\@glxtr@record}[3]{}%
278 \let\@do@wrglossary\glxtr@do@wrglossary
279 \let\@glxtr@saveentrycounter\glxtr@indexonly@saveentrycounter
280 \let\glxtrundefaction\@glxtr@err@undefaction
281 \let\glxtr@warnonexistsordo\@gobble
282 \let\@glxtr@recordcounter\@glxtr@noop@recordcounter
283 \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
284 \undef\glxtrsetaliasnoindex
285 }%
286 \or

Only record (don't index).
287 \def\glxtr@setup@record{%
288 \@glxtr@autoseeindexfalse
289 \let\@do@seeglossary\@glxtr@recordsee
290 \let\@glxtr@record\@glxtr@record
291 \let\@do@wrglossary\@glxtr@do@record@wrglossary
292 \let\@glxtr@saveentrycounter\relax
293 \let\glxtrundefaction\@glxtr@warn@undefaction
294 \let\glxtr@warnonexistsordo\@glxtr@warn@onexistsordo
295 \glxtr@addloclistfield
296 \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
297 \let\@glxtr@recordcounter\@glxtr@op@recordcounter
298 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%

Switch off the index suppression for aliased entries. (bib2gls will deal with them.)
299 \def\glxtrsetaliasnoindex{}%

```

`\@gls@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.

```
300 \ifdef\@gls@setupsort@none{\@gls@setupsort@none}{}%
```

Load glossaries-extra-bib2gls:

```
301 \RequirePackage{glossaries-extra-bib2gls}%
```

```
302 }%
```

```
303 \or
```

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

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

```
305 \renewcommand*{\@do@seeglossary}{\@glsxtr@dosee@alsoindex@glossary}%
```

```
306 \let\@glsxtr@record\@glsxtr@record
```

```
307 \let\@do@wrglossary\glsxtr@do@alsoindex@wrglossary
```

```
308 \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
```

```
309 \let\glsxtrundefaction\@glsxtr@warn@undefaction
```

```
310 \let\glsxtr@warnonexistssordo\@glsxtr@warn@onexistssordo
```

```
311 \glsxtr@addloclistfield
```

```
312 \let\@glsxtr@recordcounter\@glsxtr@op@recordcounter
```

```
313 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
```

```
314 \undef\glsxtrsetaliasnoindex
```

```
315 }%
```

```
316 \fi
```

```
317 }
```

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.

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

```
318 \newcommand*{\@glsxtr@docdefval}{0}
```

Need to provide conditional commands that are backward compatible:

`\ifglsxtrdocdef`

```
319 \newcommand*{\ifglsxtrdocdef}{\ifnum\@glsxtr@docdefval>0 }
```

`\glsxtrdocdeftrue`

```
320 \newcommand*{\@glsxtrdocdeftrue}{\def\@glsxtr@docdefval{1}}
```

`\glsxtrdocdeffalse`

```
321 \newcommand*{\@glsxtrdocdeffalse}{\def\@glsxtr@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.

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

```

323 [\@glsxtr@docdefsetting\@glsxtr@docdefval]%
324 {false,true,restricted}[true]%
325 {%
326   \ifnum\@glsxtr@docdefval=2\relax
327     \renewcommand*\@glsdoifexistsorwarn{\@glsdoifexists}%
328   \fi
329 }

ocdefrestricted
330 \newcommand*\@ifglsxtrdocdefrestricted{\@ifnum\@glsxtr@docdefval=2 }

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

indexcrossrefs  Automatically index cross references at the end of the document
332 \define@boolkey{glossaries-extra.sty}[\@glsxtr]{indexcrossrefs}[true]{%
333   \ifglsxtrindexcrossrefs
334   \else
335     \renewcommand*\@glsxtr@autoindexcrossrefs{}}%
336 \fi
337 }

Switch off since this can increase the build time.
338 \@glsxtrindexcrossrefsfalse

But allow see key to switch it on automatically.

oindexcrossrefs
339 \newcommand*\@glsxtr@autoindexcrossrefs{\@glsxtrindexcrossrefstrue}

autoseeindex  Provide a boolean option to allow the user to prevent the automatic indexing of the cross-
               referencing keys see, seealso and alias.
340 \define@boolkey{glossaries-extra.sty}[\@glsxtr]{autoseeindex}[true]{%
341 }
342 \@glsxtr@autoseeindextrue

iesExtraWarning  Allow users to suppress warnings.
343 \newcommand*\GlossariesExtraWarning[1]{\PackageWarning{glossaries-extra}{#1}}

raWarningNoLine  Allow users to suppress warnings.
344 \newcommand*\GlossariesExtraWarningNoLine[1]{%
345   \PackageWarningNoLine{glossaries-extra}{#1}}

346 \@glsxtr@declareoption{nowarn}{%
347   \let\GlossariesExtraWarning\@gobble
348   \let\GlossariesExtraWarningNoLine\@gobble
349   \glsxtr@doooption{nowarn}%
350 }

```

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

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

`postdot` Shortcut for `nopostdot=false`

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

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

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

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

```
366 \define@key{glossaries-extra.sty}{postpunc}{%
367   \glxtr@doption{nopostdot=false}%
368   \ifstrequal{#1}{dot}%
369   {%
370     \renewcommand*{\@glxtr@defpostpunc}{%
371       \renewcommand*{\glspostdescription}{.\spacefactor\sfcode'\. }%
372     }%
373   }%
374   {%
375     \ifstrequal{#1}{comma}%
376     {%
377       \renewcommand*{\@glxtr@defpostpunc}{%
378         \renewcommand*{\glspostdescription}{,}%
379       }%
380     }%
381     {%
382       \ifstrequal{#1}{none}%
383       {%
384         \glxtr@doption{nopostdot=true}%
385         \renewcommand*{\@glxtr@defpostpunc}{%
386           \renewcommand*{\glspostdescription}{}%
387         }%
388       }%
389     }%
390   }
```

```

390     \renewcommand*{\@glsxtr@defpostpunc}{%
391     \renewcommand*{\glspostdescription}{#1}%
392     }%
393     }%
394     }%
395     }%
396 }

```

`glsxtrabbrvtype` Glossary type for abbreviations.

```

397 \newcommand*{\glsxtrabbrvtype}{\glsdefaulttype}

```

`abbreviationsdef` Set by abbreviations option.

```

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

```

`abbreviationsdef`

```

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

   If the acronym option hasn't been used, change \acronymtype to \glsxtrabbrvtype.
409   \ifglsacronym
410   \else
411     \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
412   \fi
413 }%

```

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

```

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

```

`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`).

```

417 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
418   \newcommand*{\ab}{\cgl}%
419   \newcommand*{\abp}{\cglsp}%
420   \newcommand*{\as}{\glsxtrshort}%
421   \newcommand*{\asp}{\glsxtrshortpl}%
422   \newcommand*{\al}{\glsxtrlong}%
423   \newcommand*{\alp}{\glsxtrlongpl}%
424   \newcommand*{\af}{\glsxtrfull}%

```

```

425 \newcommand*{\afp}{\glxstrfullpl}%
426 \newcommand*{\Ab}{\cGls}%
427 \newcommand*{\Abp}{\cGlspl}%
428 \newcommand*{\As}{\Glsxtrshort}%
429 \newcommand*{\Asp}{\Glsxtrshortpl}%
430 \newcommand*{\Al}{\Glsxtrlong}%
431 \newcommand*{\Alp}{\Glsxtrlongpl}%
432 \newcommand*{\Af}{\Glsxtrfull}%
433 \newcommand*{\Afp}{\Glsxtrfullpl}%
434 \newcommand*{\AB}{\cGLS}%
435 \newcommand*{\ABP}{\cGLSpl}%
436 \newcommand*{\AS}{\GLSxtrshort}%
437 \newcommand*{\ASP}{\GLSxtrshortpl}%
438 \newcommand*{\AL}{\GLSxtrlong}%
439 \newcommand*{\ALP}{\GLSxtrlongpl}%
440 \newcommand*{\AF}{\GLSxtrfull}%
441 \newcommand*{\AFP}{\GLSxtrfullpl}%

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

  Disable this command after it's been used.
443 \let\GlsXtrDefineAbbreviationShortcuts\relax
444 }

```

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

```

445 \newcommand*{\GlsXtrDefineAcShortcuts}{%
446   \newcommand*{\ac}{\cglsl}%
447   \newcommand*{\acp}{\cglspl}%
448   \newcommand*{\acs}{\glxstrshort}%
449   \newcommand*{\acsp}{\glxstrshortpl}%
450   \newcommand*{\acl}{\glxtrlong}%
451   \newcommand*{\aclp}{\glxtrlongpl}%
452   \newcommand*{\acf}{\glxstrfull}%
453   \newcommand*{\acfp}{\glxstrfullpl}%
454   \newcommand*{\Ac}{\cGls}%
455   \newcommand*{\Acp}{\cGLSpl}%
456   \newcommand*{\Acs}{\Glsxtrshort}%
457   \newcommand*{\Acsp}{\Glsxtrshortpl}%
458   \newcommand*{\Acl}{\Glsxtrlong}%
459   \newcommand*{\Aclp}{\Glsxtrlongpl}%
460   \newcommand*{\Acf}{\Glsxtrfull}%
461   \newcommand*{\Acfp}{\Glsxtrfullpl}%
462   \newcommand*{\AC}{\cGLS}%
463   \newcommand*{\ACP}{\cGLSpl}%
464   \newcommand*{\ACS}{\GLSxtrshort}%
465   \newcommand*{\ACSP}{\GLSxtrshortpl}%
466   \newcommand*{\ACL}{\GLSxtrlong}%
467   \newcommand*{\ACLP}{\GLSxtrlongpl}%
468   \newcommand*{\ACF}{\GLSxtrfull}%

```

```

469 \newcommand*{\ACFP}{\GLSxtrfullpl}%
470 \providecommand*{\newabbr}{\newabbreviation}%

Disable this command after it's been used.
471 \let\GLSxtrDefineAcShortcuts\relax
472 }

```

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

```

473 \newcommand*{\GLSxtrDefineOtherShortcuts}{%
474 \newcommand*{\newentry}{\newglossaryentry}%
475 \ifdef\printsymbols
476 {%
477 \newcommand*{\newsym}{\glxtrnewsymbol}%
478 }{}%
479 \ifdef\printnumbers
480 {%
481 \newcommand*{\newnum}{\glxtrnewnumber}%
482 }{}%
483 \let\GLSxtrDefineOtherShortcuts\relax
484 }

```

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.

```

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

```

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

```

486 \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).

```

487 \define@choicekey{glossaries-extra.sty}{shortcuts}%
488 [\@glxtr@shortcutsval\@glxtr@shortcutsnr]%
489 {acronyms,acro,abbreviations,abbr,other,all,true,ac,none,false}[true]{%
490 \ifcase\@glxtr@shortcutsnr\relax % acronyms
491 \renewcommand*{\@glxtr@setupshortcuts}{%
492 \glacrshortcuttrue
493 \DefineAcronymSynonyms
494 }%
495 \or % acro
496 \renewcommand*{\@glxtr@setupshortcuts}{%
497 \glacrshortcuttrue

```

```

498     \DefineAcronymSynonyms
499   }%
500   \or % abbreviations
501     \renewcommand*{\@glxtr@setupshortcuts}{%
502       \GlsXtrDefineAbbreviationShortcuts
503     }%
504   \or % abbr
505     \renewcommand*{\@glxtr@setupshortcuts}{%
506       \GlsXtrDefineAbbreviationShortcuts
507     }%
508   \or % other
509     \renewcommand*{\@glxtr@setupshortcuts}{%
510       \GlsXtrDefineOtherShortcuts
511     }%
512   \or % all
513     \renewcommand*{\@glxtr@setupshortcuts}{%
514       \glsacrshortcutstrue
515       \GlsXtrDefineAcShortcuts
516       \GlsXtrDefineAbbreviationShortcuts
517       \GlsXtrDefineOtherShortcuts
518     }%
519   \or % true
520     \renewcommand*{\@glxtr@setupshortcuts}{%
521       \glsacrshortcutstrue
522       \GlsXtrDefineAcShortcuts
523       \GlsXtrDefineAbbreviationShortcuts
524       \GlsXtrDefineOtherShortcuts
525     }%
526   \or % ac
527     \renewcommand*{\@glxtr@setupshortcuts}{%
528       \glsacrshortcutstrue
529       \GlsXtrDefineAcShortcuts
530     }%
531   \else % none, false
532     \renewcommand*{\@glxtr@setupshortcuts}{}%
533   \fi
534 }

```

Leave none and false as last option.

`lsxtr@doaccsupp`

```
535 \newcommand*{\@glxtr@doaccsupp}{}

```

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

```

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

```

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

```
538 \newcommand{\glstrNoGlossaryWarning}[1]{%
539   \@glstr@defaultnoglossarywarning{#1}%
540 }
```

nomissingglstext If true, suppress the text produced if the external glossary file is missing.

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

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

glstr@redefstyles

```
552 \newcommand*{\@glstr@redefstyles}{}
```

stylemods

```
553 \define@key{glossaries-extra.sty}{stylemods}[default]{%
554   \ifstrequal{#1}{default}%
555   {%
556     \renewcommand*{\@glstr@redefstyles}{%
557       \RequirePackage{glossaries-extra-stylemods}}%
558   }%
559   {%
560     \ifstrequal{#1}{all}%
561     {%
562       \renewcommand*{\@glstr@redefstyles}{%
563         \PassOptionsToPackage{all}{glossaries-extra-stylemods}%
564         \RequirePackage{glossaries-extra-stylemods}%
565       }%
566     }%
567     {%
568       \renewcommand*{\@glstr@redefstyles}{}%
569       \@for\@glstr@tmp:=#1\do{%
570         \IfFileExists{glossary-\@glstr@tmp.sty}%
571         {%
572           \eappto\@glstr@redefstyles{%
573             \noexpand\RequirePackage{glossary-\@glstr@tmp}}%
574           }%
575         {%
576           \PackageError{glossaries-extra}%

```

```

577         {Glossaries style package ‘glossary- \@glsxtr@tmp.sty’
578         doesn’t exist (did you mean to use the ‘style’ key?)}%
579         {The list of values (#1) in the ‘stylemods’ key should
580         match the glossary-xxx.sty files provided with
581         glossaries.sty}%
582     }%
583 }%
584 \appto\@glsxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
585 }
586 }%
587 }

```

glsxtr@do@style

```

588 \newcommand*{\@glsxtr@do@style}{}

```

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

```

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

```

Defer actual style change:

```

590 \renewcommand*{\@glsxtr@do@style}{%

```

Set this as the default style:

```

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

```

Set this style:

```

592 \setglossarystyle{#1}%

```

```

593 }%

```

```

594 }

```

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

```

595 \newcommand*{\glsxtr@inc@wrglossaryctr}[1]{}

```

ocationHyperlink

```

\glsxtrinternallocationhyperlink{<counter>}{<prefix>}{<location>}

```

The first two arguments are always control sequences.

```

596 \newcommand*{\GlsXtrInternalLocationHyperlink}[3]{%

```

```

597 \glsxtrhyperlink{#1#2#3}{#3}%

```

```

598 }

```

cationhyperlink

```

599 \newcommand*{\@glsxtr@wrglossary@locationhyperlink}[3]{%

```

```

600 \pageref{wrglossary.#3}%

```

```

601 }

```

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

```

602 \@glsxtr@declareoption{indexcounter}{%
603   \glsxtr@doption{counter=wrglossary}%
604   \ifundef\c@wrglossary
605   {%
606     \newcounter{wrglossary}%
607     \renewcommand{\thewrglossary}{\arabic{wrglossary}}%
608   }%
609 }%
610 \renewcommand*\@glsxtr@inc@wrglossaryctr}[1]{%
611   \refstepcounter{wrglossary}%
612   \label{wrglossary.\thewrglossary}%
613 }%
614 \renewcommand*\@GlsXtrInternalLocationHyperlink}[3]{%
615   \ifdefstring\glsentrycounter{wrglossary}%
616   {%
617     \@glsxtr@wrglossary@locationhyperlink{##1}{##2}{##3}%
618   }%
619   {\glsxtrhyperlink{##1##2##3}{##3}}%
620 }%
621 }

```

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

```

622 \newcommand*\@glsxtrwrglossmark{}

```

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

```

623 \newcommand*\@@glsxtrwrglossmark{}
624 \AtBeginDocument{\renewcommand*\@glsxtrwrglossmark{\@glsxtrwrglossmark}}

```

`sxtrwrglossmark` Does nothing by default.

```

625 \newcommand*\@glsxtrwrglossmark{\ensuremath{\cdotp}}

```

`debug` Provide extra debug options.

```

626 \define@choicekey{glossaries-extra.sty}{debug}
627 [ \@glsxtr@debugval \@glsxtr@debugnr ]%
628 {true,false,showtargets,showwrgloss,all}[true]{%
629   \ifcase \@glsxtr@debugnr \relax % true
630   \glsxtr@doption{debug=true}%
631   \renewcommand*\@glsxtrwrglossmark{}%
632   \or % false
633   \glsxtr@doption{debug=false}%
634   \renewcommand*\@glsxtrwrglossmark{}%
635   \or % showtargets
636   \glsxtr@doption{debug=showtargets}%

```

```

637 \or % showwrgloss
638 \glxstr@doption{debug=true}%
639 \renewcommand*{\@glxstrwrglossmark}{\glxstrwrglossmark}%
640 \or % all
641 \glxstr@doption{debug=showtargets}%
642 \renewcommand*{\@glxstrwrglossmark}{\glxstrwrglossmark}%
643 \fi
644 }

```

Pass all other options to glossaries.

```

645 \DeclareOptionX*{%
646 \expandafter\glxstr@doption\expandafter{\CurrentOption}}

```

Process options.

```
647 \ProcessOptionsX
```

Load glossaries if not already loaded.

```
648 \RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
649 \@glxstr@doaccsupp
```

Redefine \glspostdescription if required.

```
650 \@glxstr@defpostpunc
```

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

```

651 \def\glsshowtarget#1{%
652 \glxstrtitleorpdforheading
653 {%
654 \ifmmode
655 \texttt{\small [#1]}%
656 \else
657 \ifinner
658 \texttt{\small [#1]}%
659 \else
660 \marginpar{\texttt{\small #1}}%
661 \fi
662 \fi
663 }%
664 {[#1]}%
665 {\texttt{\small [#1]}}%
666 }

```

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

```
667 \let\@glxstr@org@doseeglossary\@do@seeglossary
```

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

```

668 \newcommand*{\@glxstr@doseeglossary}[2]{%
669 \glsdofexists{#1}%

```

```

670  {%
671    \@glsxtrwrglossmark
672    \@glsxtr@org@doseeglossary{#1}{#2}%
673  }%
674 }

oindex@glossary
675 \newcommand*{\@glsxtr@dosee@alsoindex@glossary}[2]{%
676   \@glsxtr@recordsee{#1}{#2}%
677   \@glsxtr@doseeglossary{#1}{#2}%
678 }

@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.)
679 \let\@glsxtr@org@gloautosee\@glo@autosee

                  Check if user tried autoseeindex=false when it can't be supported.
680 \if@glsxtr@autoseeindex
681 \else
682   \ifdef\@glsxtr@org@gloautosee
683   {%
684     {\PackageError{glossaries-extra}{‘autoseeindex=false’ package
685       option requires at least v4.30 of glossaries.sty}%
686     {You need to update the glossaries.sty package}%
687   }
688 \fi

\@glo@autosee  If \@glo@autosee has been defined (glossaries v4.30 onwards), redefine it to test the au-
               toseeindex option.
689 \ifdef\@glo@autosee
690 {%
691   \renewcommand*{\@glo@autosee}{%
692     \if@glsxtr@autoseeindex\@glsxtr@org@gloautosee\fi}%
693 }%
694 {}

checkseeallowed  Don't prohibit the use of the see key before the indexing files have been opened if the auto-
                  matic see indexing has been disabled, since it's no longer an issue.
695 \renewcommand*{\gls@checkseeallowed}{%
696   \if@glsxtr@autoseeindex\@gls@see@noindex\fi
697 }

                  Define abbreviations glossaries if required.
698 \@glsxtr@abbreviationsdef
699 \let\@glsxtr@abbreviationsdef\relax

                  Setup shortcuts if required.
700 \@glsxtr@setupshortcuts

```

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

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

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

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

Now define the user command:

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

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

```
714 \let\glxtr@org@@do@wrglossary\@@do@wrglossary
```

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

```
715 \newcommand*{\glxtr@@do@wrglossary}[1]{%
716   \@glxtrwrglossmark
717   \glxtr@inc@wrglossaryctr{#1}%
718   \glxtr@org@@do@wrglossary{#1}%
719 }
```

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

```
720 \let\glxtr@saveentrycounter\@gl@s@saveentrycounter
```

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

```
721 \let\@gl@s@saveentrycounter\glxtr@indexonly@saveentrycounter
```

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

`sxtrdialecthook`

```
722 \newcommand*{\@glxsxtrdialecthook}{}
```

Set up record option if required.

```
723 \glxtr@setup@record
```

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

```
724 \AtBeginDocument{%
725   \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
726   \def\@glxstrundeftag{\glxstrundeftag}%
727 }
```

1.2 Extra Utilities

`\glxstrifemptyglossary`

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

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

```
728 \newcommand{\glxstrifemptyglossary}[3]{%
729   \ifcsdef{glo@list@#1}%
730   {%
731     \ifcsstring{glo@list@#1}{,}{#2}{#3}%
732   }%
733   {%
734     \glxstrundefaction{Glossary type '#1' doesn't exist}{}%
735     #2%
736   }%
737 }
```

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

```
738 \newcommand*\glxstrifkeydefined[3]{%
739   \key@ifundefined{glossentry}{#1}{#3}{#2}%
740 }
```

`\glxstrprovidestoragekey` Like `\glxstraddstoragekey` but does nothing if the key has already been defined.

```
741 \newcommand*\glxstrprovidestoragekey{%
742   \@ifstar\@sglxstr@provide@storagekey\@glxstr@provide@storagekey
743 }
```

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

```
744 \newcommand*\@glxstr@provide@storagekey[3]{%
745   \key@ifundefined{glossentry}{#1}%
746   {%
747     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
748     \appto\@glx@keymap{,}{#1}{#1}%
749     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

```

750 \appto\@newglossaryentryposthook{%
751 \letcs{\@glo@tmp}{\@glo@#1}%
752 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
753 }%

```

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

```

754 \ifblank{#3}
755 {}%
756 {%
757 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
758 }%
759 }%
760 {%

```

Provide the no-link command if not already defined.

```

761 \ifblank{#3}
762 {}%
763 {%
764 \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
765 }%
766 }%
767 }

```

`\provide@storagekey` Starred version.

```

768 \newcommand*\s@glstr@provide@storagekey}[1]{%
769 \key@ifundefined{glossentry}{#1}%
770 {%
771 \expandafter\newcommand\expandafter*\expandafter
772 {\csname gls@assign@#1@field\endcsname}[2]{%
773 \@gls@expand@field{##1}{#1}{##2}%
774 }%
775 }%
776 {}%
777 \@glstr@provide@addstoragekey{#1}%
778 }

```

The name of a text-block control sequence can be stored in a field (given by `\GlsXtrFmtField`). This command can then be used with `\glsxtrfmt[⟨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`

```

779 \newcommand{\GlsXtrFmtField}{useri}

```

`\tDefaultOptions`

```

780 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```
781 \newrobustcmd*{\glstrfmt}[3][\ifstar\s@glstrfmt\glstrfmt]
```

`\@glstrfmt` Unstarred form.

```
782 \newcommand*{\@glstrfmt}[3][\@glstrfmt{#1}{#2}{#3}]
```

`\s@glstrfmt` Starred form.

```
783 \newcommand*{\s@glstrfmt}[3][\%
784 \new@ifnextchar[\s@glstrfmt{#1}{#2}{#3}]%
785 {\@glstrfmt{#1}{#2}{#3}}%
786 }
```

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

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

`\@@glstrfmt` Actual inner working.

```
788 \newcommand*{\@@glstrfmt}[4]{%
    Since there's no post-link hook to worry about, grouping can be added to provide some protection against nesting (but in general nested link text should be avoided).
789 \begingroup
790 \def\glslabel{#2}%
791 \glsoifexistsordo{#2}%
792 {%
793 \ifglshasfield{\GlsXtrFmtField}{#2}%
794 {%
795 \let\do@glslink@checkfirsthyper\relax
796 \expandafter\@glslink\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
797 {\glstrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
798 }%
799 {\glstrfmtdisplay{@firstofone}{#3}{#4}}%
800 }%
801 {%
    Has the default noindex been counteracted? If so, this needs \glsadd in case bib2gls needs to pick up the record.
802 \begingroup
803 \@glslsetdefault@glslink@opts
804 \setkeys{glslink}{\GlsXtrFmtDefaultOptions,#1}%
805 \ifKV@glslink@noindex\else\glsadd{#2}\fi
806 \endgroup
807 \glstrfmtdisplay{@firstofone}{#3}{#4}%
808 }%
809 \endgroup
810 }
```

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

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

`\glxtrentryfmt` No link or indexing.

```
812 \ifdef\texorpdfstring
813 {
814   \newcommand*{\glxtrentryfmt}[2]{%
815     \texorpdfstring{\@glxtrentryfmt{#1}{#2}}{#2}%
816   }
817 }
818 {
819   \newcommand*{\glxtrentryfmt}{\@glxtrentryfmt}
820 }
```

`@glxtrentryfmt`

```
821 \newrobustcmd*{\@glxtrentryfmt}[2]{%
822   \glsoifexistsordo{#1}%
823   {%
824     \ifglshasfield{\GlsXtrFmtField}{#1}%
825     {%
826       \csuse{\glscurrentfieldvalue}{#2}%
827     }%
828     {#2}%
829   }%
830   {#2}%
831 }
```

`\glstrfieldlistadd` 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.

```
832 \newcommand*{\glstrfieldlistadd}[3]{%
833   \listcsadd{glo@\glsetoklabel{#1}@#2}{#3}%
834 }
```

`\glstrfieldlistgadd` Similarly but uses `\listcsgadd`.

```
835 \newcommand*{\glstrfieldlistgadd}[3]{%
836   \listcsgadd{glo@\glsetoklabel{#1}@#2}{#3}%
837 }
```

`\glstrfieldliststeadd` Similarly but uses `\listcseadd`.

```
838 \newcommand*{\glstrfieldliststeadd}[3]{%
839   \listcseadd{glo@\glsetoklabel{#1}@#2}{#3}%
840 }
```

`\glstrfieldlistxadd` Similarly but uses `\listcsxadd`.

```
841 \newcommand*{\glstrfieldlistxadd}[3]{%
```

```
842 \listcsxadd{glo@\glsdetoklabel{#1}@#2}{#3}%
843 }
```

Now provide commands to iterate over these lists.

fielddolistloop

```
844 \newcommand*{\glstrfielddolistloop}[2]{%
845 \dolistcsloop{glo@\glsdetoklabel{#1}@#2}%
846 }
```

fieldforlistloop

```
847 \newcommand*{\glstrfieldforlistloop}[3]{%
848 \forlistcsloop{#3}{glo@\glsdetoklabel{#1}@#2}%
849 }
```

List element tests:

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

```
850 \newcommand*{\glstrfieldifinlist}[5]{%
851 \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
852 }
```

rfieldxifinlist Expands item.

```
853 \newcommand*{\glstrfieldxifinlist}[5]{%
854 \xifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
855 }
```

lsxtrforcsvfield

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

```
856 \newcommand*{\glsxtrforcsvfield}[3]{%
857 \@glxtrifhasfield{#2}{#1}%
858 {%
859 \let\glxxtrendfor\@endfortrue
860 \@for\@glxtr@label:=\glscurrentfieldvalue\do
861 {\expandafter#3\expandafter{\@glxtr@label}}}%
862 }%
863 }
```

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

```
864 \newrobustcmd{\glxtrifhasfield}{%
865 \@ifstar{\s@glxtrifhasfield}{\@glxtrifhasfield}%
866 }
```

`\lsxtrifhasfield` Unstarred version adds grouping.

```
867 \newcommand{\@glxtrifhasfield}[4]{%
868   {\s@glxtrifhasfield{#1}{#2}{#3}{#4}}%
869 }
```

`\lsxtrifhasfield` Starred version omits grouping.

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

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

Just uses `\ifcsundef`.

```
878 \newcommand{\GlsXtrIfFieldUndef}[2]{%
879   \ifcsundef{glo@glsdetoklabel{#2}@#1}%
880 }
```

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

```
881 \newcommand*{\glxtrusefield}[2]{%
882   \@gl@entry@field{#1}{#2}%
883 }
```

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

```
884 \newcommand*{\Glsxtrusefield}[2]{%
885   \@gl@entry@field{#1}{#2}%
886 }
```

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

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

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

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

`\setfieldifexists`

```
889 \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.

```
890 \newrobustcmd*{\GlsXtrSetField}[3]{%
```

```

891 \glstrsetfieldifexists{#1}{#2}%
892 {\csdef{glo@glstetoklabel{#1}@#2}{#3}}%
893 }

\GlsXtrLetField Uses \cslet instead. Third argument should be a macro.
894 \newrobustcmd*{\GlsXtrLetField}[3]{%
895 \glstrsetfieldifexists{#1}{#2}%
896 {\cslet{glo@glstetoklabel{#1}@#2}{#3}}%
897 }

\csGlsXtrLetField Uses \csletcs instead. Third argument should be a control sequence name.
898 \newrobustcmd*{\csGlsXtrLetField}[3]{%
899 \glstrsetfieldifexists{#1}{#2}%
900 {\csletcs{glo@glstetoklabel{#1}@#2}{#3}}%
901 }

\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.
902 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
903 \glstrsetfieldifexists{#1}{#2}%
904 {\csletcs{glo@glstetoklabel{#1}@#2}{glo@glstetoklabel{#3}@#4}}%
905 }

\gGlsXtrSetField Allow the user to set a field. First argument entry label, second argument field label, third
argument value.
906 \newrobustcmd*{\gGlsXtrSetField}[3]{%
907 \glstrsetfieldifexists{#1}{#2}%
908 {\csgdef{glo@glstetoklabel{#1}@#2}{#3}}%
909 }

\XGlsXtrSetField
910 \newrobustcmd*{\XGlsXtrSetField}[3]{%
911 \glstrsetfieldifexists{#1}{#2}%
912 {\protected@csxdef{glo@glstetoklabel{#1}@#2}{#3}}%
913 }

\eGlsXtrSetField
914 \newrobustcmd*{\eGlsXtrSetField}[3]{%
915 \glstrsetfieldifexists{#1}{#2}%
916 {\protected@csedef{glo@glstetoklabel{#1}@#2}{#3}}%
917 }

\XtrIfFieldEqStr
918 \newrobustcmd*{\GlsXtrIfFieldEqStr}[5]{%
919 \glstrifhasfield{#1}{#2}%
920 {%
921 \ifdefstring{glscurrentfieldvalue}{#3}{#4}{#5}%
922 }%

```

```

923   {#5}%
924 }

```

`\glxtrpageref` Like `\glsrefentry` but references the page number instead (if entry counting is on).

```

925 \ifglentrycounter
926   \newcommand*{\glxtrpageref}[1]{\pageref{glsentry-\glsdetoklabel{#1}}}
927 \else
928   \ifglssubentrycounter
929     \newcommand*{\glxtrpageref}[1]{\pageref{glsentry-\glsdetoklabel{#1}}}
930   \else
931     \newcommand*{\glxtrpageref}[1]{\gls{#1}}
932   \fi
933 \fi

```

`lossarypreamble`

```

934 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
935   \ifcsdef{glolist@#1}%
936   {%
937     \ifcsundef{@glossarypreamble@#1}%
938     {\csdef{@glossarypreamble@#1}{}}%
939   }%
940   \csappto{@glossarypreamble@#1}{#2}%
941 }%
942 {%
943   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
944 }%
945 }

```

`lossarypreamble`

```

946 \newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
947   \ifcsdef{glolist@#1}%
948   {%
949     \ifcsundef{@glossarypreamble@#1}%
950     {\csdef{@glossarypreamble@#1}{}}%
951   }%
952   \cspreto{@glossarypreamble@#1}{#2}%
953 }%
954 {%
955   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
956 }%
957 }

```

1.3 Modifications to Commands Provided by glossaries

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

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

`ewglossaryentry`

```
958 \renewcommand*{\longnewglossaryentry}{%
959   \@ifstar\@glstr@s@longnewglossaryentry\@glstr@longnewglossaryentry
960 }
```

`ewglossaryentry`

Starred version.

```
961 \newcommand{\@glstr@s@longnewglossaryentry}[3]{%
962   \glsdofnoexists{#1}%
963   {%
964     \bgroup
965     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
966     \long\def\@newglossaryentryprehook{%
967       \long\def\@glo@desc{#3}%
968       \@org@newglossaryentryprehook
969     }%
970     \renewcommand*{\gl@assign@desc}[1]{%
971       \global\cslet{glo@\glsetoklabel{#1}@desc}{\@glo@desc}%
972       \global\cslet{glo@\glsetoklabel{#1}@descplural}{\@glo@descplural}%
973     }
974     \gl@defglossaryentry{#1}{#2}%
975   \egroup
976 }%
977 }
```

`ewglossaryentry`

Unstarred version.

```
978 \newcommand{\@glstr@longnewglossaryentry}[3]{%
979   \glsdofnoexists{#1}%
980   {%
981     \bgroup
982     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
983     \long\def\@newglossaryentryprehook{%
984       \long\def\@glo@desc{#3\glstrpostlongdescription}%
985       \@org@newglossaryentryprehook
986     }%
987     \renewcommand*{\gl@assign@desc}[1]{%
988       \global\cslet{glo@\glsetoklabel{#1}@desc}{\@glo@desc}%
989       \global\cslet{glo@\glsetoklabel{#1}@descplural}{\@glo@descplural}%
990     }
991     \gl@defglossaryentry{#1}{#2}%
992   \egroup
993 }%
994 }
```

The following is different from the base glossaries.sty:

```
989   \global\cslet{glo@\glsetoklabel{#1}@descplural}{\@glo@descplural}%
990   }
991   \gl@defglossaryentry{#1}{#2}%
992 \egroup
993 }%
994 }
```

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

```
995 \newcommand*{\glxtrpostlongdescription}{\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.

```
996 \renewcommand{\newignoredglossary}{%
997 \ifstar\glxtr@s@newignoredglossary\glxtr@org@newignoredglossary
998 }
```

ignoredglossary The original definition is patched to check for existence.

```
999 \newcommand*{\glxtr@org@newignoredglossary}[1]{%
1000 \ifcsdef{glolist@#1}
1001 {%
1002 \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
1003 }%
1004 {%
1005 \ifdefempty\@ignored@glossaries
1006 {%
1007 \edef\@ignored@glossaries{#1}%
1008 }%
1009 {%
1010 \eappto\@ignored@glossaries{,#1}%
1011 }%
1012 \csgdef{glolist@#1}{,}%
1013 \ifcsundef{gls@#1@entryfmt}%
1014 {%
1015 \defglsentryfmt[#1]{\glsentryfmt}%
1016 }%
1017 {}%
1018 \ifdefempty\@gls@nohyperlist
1019 {%
1020 \renewcommand*{\@gls@nohyperlist}{#1}%
1021 }%
1022 {%
1023 \eappto\@gls@nohyperlist{,#1}%
1024 }%
1025 }%
1026 }
```

ignoredglossary Starred form.

```
1027 \newcommand*{\glxtr@s@newignoredglossary}[1]{%
1028 \ifcsdef{glolist@#1}
1029 {%
1030 \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
1031 }%
1032 {%
1033 \ifdefempty\@ignored@glossaries
```

```

1034   {%
1035     \edef\@ignored@glossaries{#1}%
1036   }%
1037   {%
1038     \eappto\@ignored@glossaries{,#1}%
1039   }%
1040   \csgdef{glolist@#1}{,}%
1041   \ifcsundef{gls@#1@entryfmt}%
1042   {%
1043     \defglsentryfmt[#1]{\glsentryfmt}%
1044   }%
1045   {}%
1046 }%
1047 }

```

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

```

1048 \glsifusetranslator
1049 {%
1050   \renewcommand*{\glsettoctitle}[1]{%
1051     \ifcsdef{gls@tr@set@#1@toctitle}%
1052     {%
1053       \csuse{gls@tr@set@#1@toctitle}%
1054     }%
1055     {%
1056       \ifcsdef{@glotype@#1@title}%
1057       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1058       {\def\glossarytoctitle{\glossarytitle}}%
1059     }%
1060   }%
1061 }
1062 {
1063   \renewcommand*{\glsettoctitle}[1]{%
1064     \ifcsdef{@glotype@#1@title}%
1065     {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1066     {\def\glossarytoctitle{\glossarytitle}}%
1067   }
1068 }

```

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

```

1069 \newcommand{\provideignoredglossary}{%
1070   \@ifstar\glxtr@s@provideignoredglossary\glxtr@provideignoredglossary
1071 }

```

`ignoredglossary` Unstarred version.

```

1072 \newcommand*{\glxtr@provideignoredglossary}[1]{%
1073   \ifcsdef{glolist@#1}
1074   {}%
1075   {%

```

```

1076 \ifdefempty\@ignored@glossaries
1077 {%
1078   \edef\@ignored@glossaries{#1}%
1079   }%
1080   {%
1081     \eappto\@ignored@glossaries{,#1}%
1082     }%
1083   \csgdef{glolist@#1}{,}%
1084   \ifcsundef{gls@#1@entryfmt}%
1085   {%
1086     \defglsentryfmt[#1]{\glsentryfmt}%
1087     }%
1088     {}%
1089   \ifdefempty\@gls@nohyperlist
1090   {%
1091     \renewcommand*{\@gls@nohyperlist}{#1}%
1092     }%
1093     {%
1094       \eappto\@gls@nohyperlist{,#1}%
1095       }%
1096   }%
1097 }

```

`ignoredglossary` Starred form.

```

1098 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
1099   \ifcsdef{glolist@#1}
1100   {%
1101     {%
1102       \ifdefempty\@ignored@glossaries
1103       {%
1104         \edef\@ignored@glossaries{#1}%
1105         }%
1106         {%
1107           \eappto\@ignored@glossaries{,#1}%
1108           }%
1109         \csgdef{glolist@#1}{,}%
1110         \ifcsundef{gls@#1@entryfmt}%
1111         {%
1112           \defglsentryfmt[#1]{\glsentryfmt}%
1113           }%
1114           {}%
1115         }%
1116 }

```

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

```

1117 \newcommand*{\glxtrcopytoglossary}[2]{%
1118   \glstoifexists{#1}%
1119   {%

```

```

1120 \ifcsdef{glolist@#2}
1121 {%
1122 \cseappto{glolist@#2}{#1,}%
1123 }%
1124 {%
1125 \glstrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
1126 }%
1127 }%
1128 }

```

1.3.1 Existence Checks

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

```

1129 \renewcommand{\glstoifexists}[2]{%
1130 \ifglstryexists{#1}{#2}%
1131 {%

```

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

```

1132 \edef\glslabel{\glstetoklabel{#1}}%
1133 \glstrundefaction{Glossary entry ‘\glslabel’
1134 has not been defined}{You need to define a glossary entry before
1135 you can reference it.}%
1136 }%
1137 }

```

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

```

1138 \renewcommand{\glstoifnoexists}[2]{%
1139 \ifglstryexists{#1}{%
1140 \glstrundefaction{Glossary entry ‘\glstetoklabel{#1}’
1141 has already been defined}{}}{#2}%
1142 }

```

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

```

1143 \ifdef\glstoifexistsordo
1144 {%
1145 \renewcommand{\glstoifexistsordo}[3]{%
1146 \ifglstryexists{#1}{#2}%
1147 {%
1148 \glstrundefaction{Glossary entry ‘\glstetoklabel{#1}’
1149 has not been defined}{You need to define a glossary entry
1150 before you can use it.}%
1151 #3%
1152 }%
1153 }%
1154 }
1155 {%
1156 \glstr@warnonexistsordo\glstoifexistsordo

```

```

1157 \newcommand{\glsdoifexistsordo}[3]{%
1158   \ifglsentryexists{#1}{#2}%
1159   {%
1160     \glstrundefaction{Glossary entry ‘\glsdetoklabel{#1}’
1161     has not been defined}{You need to define a glossary entry
1162     before you can use it.}%
1163     #3%
1164   }%
1165 }%
1166 }

```

`arynoexistsordo` Similarly for `\doifglossarynoexistsordo`.

```

1167 \ifdef\doifglossarynoexistsordo
1168 {%
1169   \renewcommand{\doifglossarynoexistsordo}[3]{%
1170     \ifglossaryexists{#1}%
1171     {%
1172       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1173       #3%
1174     }%
1175     {#2}%
1176   }%
1177 }
1178 {%
1179   \glstr@warnonexistsordo\doifglossarynoexistsordo
1180   \newcommand{\doifglossarynoexistsordo}[3]{%
1181     \ifglossaryexists{#1}%
1182     {%
1183       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1184       #3%
1185     }%
1186     {#2}%
1187   }%
1188 }
1189

```

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.

```

1190 \appto\newglossaryentryposthook{%
1191   \ifdefvoid\@glo@see
1192   {\csxdef{glo@\@glo@label @see}{}%
1193   {%
1194     \csxdef{glo@\@glo@label @see}{\@glo@see}%
1195     \if@glstr@autoseeindex
1196       \@glstr@autoindexcrossrefs
1197     \fi

```

```

1198     }%
1199 }
1200 \appto\@gls@keymap{,{see}{see}}

\glxtrusesee Apply \glsseeformat to the see key if not empty.
1201 \newcommand*{\glxtrusesee}[1]{%
1202   \glsdoifexists{#1}%
1203   {%
1204     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
1205     \ifdefempty\@glo@see
1206     }{%
1207     }%
1208     \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
1209   }%
1210 }%
1211 }

\glxtr@usesee
1212 \newcommand*{\glxtr@usesee}[1][\seename]{%
1213   \@glxtr@usesee[#1]%
1214 }

\@glxtr@usesee
1215 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
1216   \glxtruseseeformat{#1}{#2}%
1217 }

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.
1218 \newcommand*{\glxtruseseeformat}[2]{%
1219   \glsseeformat{#1}{#2}{}%
1220 }

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.
1221 \renewcommand*{\glsseeitemformat}[1]{%
1222   \ifglshashshort{\glslabel}{\glsaccesstext{#1}}{\glsaccessname{#1}}%
1223 }

lsxtruseseealso Apply \glsseeformat to the seealso key if not empty. There's no optional tag to worry about
here.
1224 \newcommand*{\glxtruseseealso}[1]{%
1225   \glsdoifexists{#1}%
1226   {%
1227     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%

```

```

1228 \ifdefempty\@glo@see
1229 {}%
1230 {%
1231 \expandafter\glxtruseseealsoformat\expandafter{\@glo@see}%
1232 }%
1233 }%
1234 }

```

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

```

1235 \newcommand*{\glxtruseseealsoformat}[1]{%
1236 \glseeformat[\seealso]{#1}{}%
1237 }

```

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

```

1238 \newrobustcmd{\glxtrseelist}[1]{%
1239 \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
1240 }

```

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

```

1241 \providecommand{\seealso}{see also}

```

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

```

1242 \ifdef\@xdycrossrefhook
1243 {
    Add the cross-reference class definition to the hook.
1244 \appto\@xdycrossrefhook{%
1245 \write\glswrite{(define-crossref-class \string"seealso\string"
1246 :unverified )}%
1247 \write\glswrite{(markup-crossref-list
1248 :class \string"seealso\string"^^J\space\space\space
1249 :open \string"\string\glxtruseseealsoformat\glsoopenbrace\string"
1250 :close \string"\glsclosebrace\string")}%
1251 }

```

Append to class list.

```

1252 \appto\@xdylocationclassorder{\space\string"seealso\string"}

```

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

```

1253 \newrobustcmd*{\glxtrindexseealso}[2]{%
1254 \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
1255 \@glxtr@recordsee{#1}{#2}%
1256 \fi
1257 \glsoifexists{#1}%

```

```

1258   {%
1259     \@glsxtrwrglossmark
1260     \def\@gls@xref{#2}%
1261     \@onelevel@sanitize\@gls@xref
1262     \@gls@checkmkidxchars\@gls@xref
1263     \gls@glossary{\csname glo@#1@type\endcsname}{%
1264       (indexentry
1265         :tkey (\csname glo@#1@index\endcsname)
1266         :xref (\string"\@gls@xref\string")
1267         :attr \string"seealso\string"
1268       )
1269     }%
1270   }%
1271 }
1272 }
1273 {
  xindy not in use or glossaries version too old to support this.
1274 \newrobustcmd*{\glsxtrindexseealso}{\glssee[\seealsoname]}
1275 }

```

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`.

```

1276 \ifdef\gls@set@xr@key
1277 {

```

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

```

1278 \define@key{glossentry}{alias}{%
1279   \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1280 }
1281 \define@key{glossentry}{seealso}{%
1282   \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1283 }

```

Add to the key mappings.

```

1284 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}

```

Set the default value.

```

1285 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%

```

Assign the field values.

```

1286 \appto\@newglossaryentryposthook{%
1287   \ifdefvoid\@glo@seealso
1288     {\csxdef{glo@\@glo@label @seealso}{}}%
1289     {%
1290       \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%

```

```

1291     \if@glstr@autoseeindex
1292     \@glstr@autoindexcrossrefs
1293     \fi
1294 }%

```

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

```

1295     \ifdefvoid\@glo@alias
1296     {\csxdef{glo@\@glo@label @alias}{}}%
1297     {%
1298     \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1299     }%
1300 }

```

Provide user-level commands to access the values.

`\glstralias`

```

1301 \newcommand*{\glstralias}[1]{\@gls@entry@field{#1}{alias}}

```

`trseealsolabels`

```

1302 \newcommand*{\glstrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}

```

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

```

1303 \appto\@glo@autoseehook{%
1304     \ifdefvoid\@glo@alias
1305     {%
1306     \ifdefvoid\@glo@seealso
1307     }%
1308     {%
1309     \edef\@do@glsee{\noexpand\glstrindexseealso
1310     {\@glo@label}{\@glo@seealso}}%
1311     \@do@glsee
1312     }%
1313     }%
1314     {%

```

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

```

1315     \ifdefvoid\@glo@see
1316     {%
1317     \edef\@do@glsee{\noexpand\glsee{\@glo@label}{\@glo@alias}}%
1318     \@do@glsee
1319     }%
1320     }%
1321     }%
1322 }%
1323 }
1324 {

```

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

`\glstralias`

```

1325 \glsaddstoragekey*{alias}{}{\glstralias}

```

trseealsolabels

```
1326 \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.

```
1327 \appto\@newglossaryentryposthook{%
1328   \ifcsvoid{glo@\@glo@label @alias}%
1329   {%
1330     \ifcsvoid{glo@\@glo@label @seealso}%
1331     {}%
1332     {%
1333       \edef\@do@glsssee{\noexpand\glsxtrindexseealso
1334         {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1335       \@do@glsssee
1336     }%
1337   }%
1338   {%
```

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

```
1339   \ifdefvoid\@glo@see
1340   {%
1341     \edef\@do@glsssee{\noexpand\glsssee
1342       {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1343     \@do@glsssee
1344   }%
1345   {}%
1346 }%
1347 }
1348 }
```

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

```
1349 \AtEndDocument{\if\glsxtrindexcrossrefs\glsxtraddallcrossrefs\fi}
```

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

```
1350 \newcommand*\glsxtraddallcrossrefs{%
1351   \forallglossaries{\@glo@type}%
1352   {%
1353     \forglsentries[\@glo@type]{\@glo@label}%
1354     {%
1355       \ifglssused{\@glo@label}%
1356       {\expandafter\@glsxtr@addunusedxrefs\expandafter{\@glo@label}}}%
1357   }%
1358 }%
1359 }
```

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

```

1360 \newcommand*{\@glsxtr@addunusedxrefs}[1]{%
1361   \letcs{\@glo@see}{glo@glstdetoklabel{#1}@see}%
1362   \ifdefvoid\@glo@see
1363   {}%
1364   {%
1365     \expandafter\glsxtr@addunused\@glo@see\@end@glsxtr@addunused
1366   }%
1367   \letcs{\@glo@see}{glo@glstdetoklabel{#1}@seealso}%
1368   \ifdefvoid\@glo@see
1369   {}%
1370   {%
1371     \expandafter\glsxtr@addunused\@glo@see\@end@glsxtr@addunused
1372   }%
1373 }
```

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

```

1374 \newcommand*{\glsxtr@addunused}[1][]{%
1375   \@glsxtr@addunused
1376 }
```

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

```

1377 \def\@glsxtr@addunused#1\@end@glsxtr@addunused{%
1378   \@for\@glsxtr@label:=#1\do
1379   {%
1380     \ifglsused{\@glsxtr@label}{}%
1381     {%
1382       \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
1383       \glsunset{\@glsxtr@label}%
1384       \expandafter\@glsxtr@addunusedxrefs\expandafter{\@glsxtr@label}%
1385     }%
1386   }%
1387 }
```

`xtrunusedformat`

```

1388 \newcommand*{\glsxtrunusedformat}[1]{\unskip}
```

1.3.2 Document Definitions

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

```

1389 \ifdef\gls@begindocdefs
1390 {%
1391   \renewcommand*{\gls@begindocdefs}{%
1392     \ifnum\@glsxtr@docdefval=1\relax
1393       \@gls@enablesavenonumberlist
```

```

1394     \edef\@gls@restoreat{%
1395         \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1396     \makeatletter
1397     \InputIfFileExists{\jobname.glsdefs}{-}{-}%
1398     \@gls@restoreat
1399     \undef\@gls@restoreat
1400     \gls@defdocnewglossaryentry
1401 \fi
1402 }
1403 }
1404 {}

```

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

```

1405 \let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
1406 \renewcommand{\makenoidxglossaries}{%
1407     \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
1408     {%
1409         \glsxtr@orgmakenoidxglossaries

```

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

```

1410     \renewcommand{\do@seeglossary}[2]{%
1411         \@glsxtrwrglossmark
1412         \edef\@gls@label{\glsdetoklabel{##1}}%
1413         \protected@write\@auxout{}{%
1414             \string\@gls@reference
1415             {\csname glo@\@gls@label @type\endcsname}%
1416             {\@gls@label}%
1417             {%
1418                 \string\glsseeformat##2}%
1419             }%
1420         }%
1421     }%

```

Check for `docdefs=restricted`:

```

1422     \if@glsxtrdocdefrestricted

```

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

```

1423     \renewcommand*{\@gls@reference}[3]{%
1424         \ifcsundef{\glsref@##1}{\csgdef{\glsref@##1}{-}}{}%
1425         \ifinlistcs{##2}{\glsref@##1}%
1426         {}%
1427         {\listcsgadd{\glsref@##1}{##2}}%
1428         \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
1429         {\csgdef{glo@\glsdetoklabel{##2}@loclist}{-}}%
1430         {}%
1431         \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
1432     }%

```

```

1433     \else
      Disable document definitions.
1434     \@glxtrdocdeffalse
1435     \fi
1436     \disable@keys{glossaries-extra.sty}{docdef}%
1437   }%
1438   {%
1439     \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1440       not permitted\MessageBreak
1441       with record=\@glxtr@record@setting\space package option}%
1442     {You may only use \string\makenoidxglossaries\space with the
1443       record=off option}%
1444   }%
1445 }

```

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

```

1446 \renewcommand*{\gls@defdocnewglossaryentry}{%
1447   \ifcase\@glxtr@docdefval
      docdef=false:
1448     \renewcommand*{\newglossaryentry}[2]{%
1449       \PackageError{glossaries-extra}{Glossary entries must
1450         be \MessageBreak defined in the preamble with \MessageBreak
1451         package option ‘docdef=false’\MessageBreak(consider using
1452         ‘docdef=restricted’)}{Move your glossary definitions to
1453         the preamble. You can also put them in a \MessageBreak separate file
1454         and load them with \string\loadglsentries.}%
1455     }%
1456   \or
      (docdef=true case.) Since the see value is now saved in a field, it can be used by entries that
      have been defined in the document.
1457     \let\gls@checkseeallowed\relax
1458     \let\newglossaryentry\new@glossaryentry
1459   \or
      Restricted mode just needs to allow the see value.
1460     \let\gls@checkseeallowed\relax
1461   \fi
1462 }%

```

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`

```

1463 \newcommand*{\GlsXtrEnableOnTheFly}{%
1464   \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1465 }

```

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

```

1466 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1467   \renewcommand*{\glsdetoklabel}[1]{%
1468     \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1469     {%
1470       \expandafter\detokenize\expandafter{##1}%
1471     }%
1472     {\detokenize{##1}}%
1473   }%
1474   \@GlsXtrEnableOnTheFly
1475 }
1476 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1477   \expandafter\if\glsbackslash#1%
1478     #3%
1479   \else
1480     #4%
1481   \fi
1482 }

```

`\GlsXtrStarflywarn`

```

1483 \newcommand*{\GlsXtrStarflywarn}{%
1484   \GlossariesExtraWarning{Experimental starred version of
1485     \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1486     read the warnings in the glossaries-extra user manual)}}%
1487 }

```

`\GlsXtrEnableOnTheFly`

```

1488 \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`

```

1489   \newcommand*{\glsxtrcat}{general}

```

`\glsxtr`

```

1490   \newcommand*{\glsxtr}[1] [] {%
1491     \def\glsxtr@keylist{##1}%
1492     \@glsxtr
1493   }

```

`\@glsxtr`

```

1494   \newcommand*{\@glsxtr}[2] [] {%
1495     \ifglsentryexists{##2}%

```

```

1496   {%
1497     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1498   }%
1499   {%
1500     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1501       description={\nopostdesc},##1}%
1502   }%
1503   \expandafter\gls\expandafter[\glstr@keylist]{##2}%
1504 }

```

\Glsxtr

```

1505 \newcommand*{\Glsxtr}[1] [] {%
1506   \def\glstr@keylist{##1}%
1507   \@Glsxtr
1508 }

```

\@Glsxtr

```

1509 \newcommand*{\@Glsxtr}[2] [] {%
1510   \ifglstryexists{##2}%
1511   {%
1512     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1513   }%
1514   {%
1515     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1516       description={\nopostdesc},##1}%
1517   }%
1518   \expandafter\gls\expandafter[\glstr@keylist]{##2}%
1519 }

```

\glstrpl

```

1520 \newcommand*{\glstrpl}[1] [] {%
1521   \def\glstr@keylist{##1}%
1522   \@glstrpl
1523 }

```

\@glstrpl

```

1524 \newcommand*{\@glstrpl}[2] [] {%
1525   \ifglstryexists{##2}%
1526   {%
1527     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1528   }%
1529   {%
1530     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1531       description={\nopostdesc},##1}%
1532   }%
1533   \expandafter\glsp\expandafter[\glstr@keylist]{##2}%
1534 }

```

\Glsxtrpl

```

1535 \newcommand*{\Glsxtrpl}[1][]{%
1536   \def\glxtr@keylist{##1}%
1537   \@Glsxtrpl
1538 }

```

\@Glsxtrpl

```

1539 \newcommand*{\@Glsxtrpl}[2][]{%
1540   \ifglstryentryexists{##2}
1541   {%
1542     \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
1543   }%
1544   {%
1545     \gls@defglossaryentry{##2}{name={##2},category=\glxtrcat,
1546       description={\nopostdesc},##1}%
1547   }%
1548   \expandafter\Glspl\expandafter[\glxtr@keylist]{##2}%
1549 }

```

\GlsXtrWarning

```

1550 \newcommand*{\GlsXtrWarning}[2]{%
1551   \def\@glxtr@optlist{##1}%
1552   \@onelevel@sanitize\@glxtr@optlist
1553   \GlossariesExtraWarning{The options '\@glxtr@optlist' have
1554     been ignored for entry '##2' as it has already been defined}%
1555 }

```

Disable commands after the glossary:

```

1556 \renewcommand\@printglossary[2]{%
1557   \def\@glxtr@printglossopts{##1}%
1558   \@glxtr@orgprintglossary{##1}{##2}%
1559   \def\@glxtr{\@glxtr@disabledflycommand\glxtr}%
1560   \def\@glxtrpl{\@glxtr@disabledflycommand\glxtrpl}%
1561   \def\@Glsxtr{\@glxtr@disabledflycommand\Glsxtr}%
1562   \def\@Glsxtrpl{\@glxtr@disabledflycommand\Glsxtrpl}%
1563 }

```

abledflycommand

```

1564 \newcommand*{\@glxtr@disabledflycommand}[1]{%
1565   \PackageError{glossaries-extra}%
1566   {\string##1\space can't be used after any of the \MessageBreak
1567     glossaries have been displayed}%
1568   {The on-the-fly commands enabled by
1569     \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1570     before the glossaries. If you want to use any entries \MessageBreak
1571     after any of the glossaries, you must use the standard \MessageBreak
1572     method of first defining the entry and then using the \MessageBreak
1573     entry with commands like \string\gls}%
1574   \@@glxtr@disabledflycommand
1575 }%

```

```
1576 \newcommand*{\@glxtr@disabledflycommand}[2][\@##2]
```

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

```
1577 \let\GlsXtrEnableOnTheFly\relax
1578 }
1579 \@onlypreamble\GlsXtrEnableOnTheFly
```

1.3.3 Existing Glossary Style Modifications

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

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

```
1580 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}
```

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

`\setglossarystyle`

```
1581 \renewcommand*{\setglossarystyle}[1]{%
1582   \ifcsundef{@glsstyle@#1}%
1583   {%
1584     \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
1585   }%
1586   {%
1587     \csname @glsstyle@#1\endcsname
```

Only set the current style if it exists.

```
1588   \protected@edef\@glxtr@current@style{#1}%
1589   }%
1590   \ifx\@glossary@default@style\relax
1591     \protected@edef\@glossary@default@style{#1}%
1592   \fi
1593 }
```

In case we have an old version of glossaries:

```
1594 \ifdef\@glossary@default@style
1595 {}
1596 {%
1597   \let\@glossary@default@style\relax
1598 }
```

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

```
1599 \ifdef\glslistdottedwidth
1600 {%
1601   \ifdim\glslistdottedwidth=.5\hsize
1602     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
1603     \AtBeginDocument{%
1604       \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
```

```

1605     \setlength{\glslistdottedwidth}{.5\columnwidth}%
1606     \fi
1607 }%
1608 \fi
1609 }
1610 {}%

```

Similarly for \glsdescwidth:

\glsdescwidth

```

1611 \ifdef\glsdescwidth
1612 {%
1613   \ifdim\glsdescwidth=.6\hsize
1614     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
1615     \AtBeginDocument{%
1616       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
1617         \setlength{\glsdescwidth}{.6\columnwidth}%
1618       \fi
1619     }%
1620   \fi
1621 }
1622 {}%

```

and for \glspagelistwidth:

\glspagelistwidth

```

1623 \ifdef\glspagelistwidth
1624 {%
1625   \ifdim\glspagelistwidth=.1\hsize
1626     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
1627     \AtBeginDocument{%
1628       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
1629         \setlength{\glspagelistwidth}{.1\columnwidth}%
1630       \fi
1631     }%
1632   \fi
1633 }
1634 {}%

```

aryentrynumbers Has the nonumberlist option been used?

```

1635 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
1636 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
1637   \glsnonumberlistfalse
1638   \renewcommand*{\glossaryentrynumbers}[1]{%
1639     \ifglstryentryexists{\glscurrententrylabel}%
1640     {%
1641       \@glsxtrpreloctag
1642       \GlsXtrFormatLocationList{#1}%
1643       \@glsxtrpostloctag
1644       \gls@save@numberlist{#1}%

```

```

1645     }{}%
1646   }%
1647 \else
1648   \glsnonumberlisttrue
1649   \renewcommand*{\glossaryentrynumbers}[1]{%
1650     \ifglsentryexists{\glscurrententrylabel}%
1651     {%
1652       \gls@save@numberlist{#1}%
1653     }{}%
1654   }%
1655 \fi

```

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

```

1656 \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`

```

1657 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
1658   \let\@glsxtrpreloctag\@glsxtrpreloctag
1659   \let\@glsxtrpostloctag\@glsxtrpostloctag
1660   \renewcommand*{\@glsxtr@pagetag}{#1}%
1661   \renewcommand*{\@glsxtr@pagetag}{#2}%
1662   \renewcommand*{\@glsxtr@savepreloctag}[2]{%
1663     \csgdef{\@glsxtr@preloctag@##1}{##2}%
1664   }%
1665   \renewcommand*{\@glsxtr@doloctag}{%
1666     \ifcsundef{\@glsxtr@preloctag\@glscurrententrylabel}%
1667     {%
1668       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
1669         Rerun required}%
1670     }%
1671     {%
1672       \csuse{\@glsxtr@preloctag\@glscurrententrylabel}%
1673     }%
1674   }%
1675 }
1676 \@onlypreamble\GlsXtrEnablePreLocationTag

```

`\glsxtrpreloctag`

```

1677 \newcommand*{\@glsxtrpreloctag}{%
1678   \let\@glsxtr@org@delimN\delimN
1679   \let\@glsxtr@org@delimR\delimR
1680   \let\@glsxtr@org@glsignore\glsignore

```

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

```

1681 \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1682 \renewcommand*{\delimN}{\%
1683 \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1684 \@glsxtr@org@delimN}%
1685 \renewcommand*{\delimR}{\%
1686 \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1687 \@glsxtr@org@delimR}%
1688 \renewcommand*{\glsignore}[1]{\%
1689 \gdef\@glsxtr@thisloctag{\relax}%
1690 \@glsxtr@org@glsignore{##1}}%
1691 \@glsxtr@doloctag
1692 }

```

glsxtrpreloctag

```

1693 \newcommand*{\@glsxtrpreloctag}{}

```

@glsxtr@pagetag

```

1694 \newcommand*{\@glsxtr@pagetag}{}%

```

glsxtr@pagetag

```

1695 \newcommand*{\@glsxtr@pagetag}{}%

```

lsxtrpostloctag

```

1696 \newcommand*{\@@glsxtrpostloctag}{\%
1697 \let\delimN\@glsxtr@org@delimN
1698 \let\delimR\@glsxtr@org@delimR
1699 \let\glsignore\@glsxtr@org@glsignore
1700 \protected@write\@auxout{}{\%
1701 {\string\@glsxtr@savepreloctag{\glscurrententrylabel}{\@glsxtr@thisloctag}}%
1702 }

```

lsxtrpostloctag

```

1703 \newcommand*{\@glsxtrpostloctag}{}

```

lsxtr@preloctag

```

1704 \newcommand*{\@glsxtr@savepreloctag}[2]{\%
1705 \protected@write\@auxout{}{\%
1706 \string\providecommand\string\@glsxtr@savepreloctag[2]}%

```

glsxtr@doloctag

```

1707 \newcommand*{\@glsxtr@doloctag}{}

```

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

```

1708 \renewcommand*{\KV@printgloss@nonumberlist}[1]{\%
1709 \XKV@plfalse
1710 \XKV@sttrue

```

```

1711 \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
1712 {%
1713   \csname glsnonumberlist\XKV@resa\endcsname
1714   \ifglsnonumberlist
1715     \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
1716   \else
1717     \def\glossaryentrynumbers##1{%
1718       \@glstrpreloctag
1719       \GlsXtrFormatLocationList{##1}%
1720       \@glstrpostloctag
1721       \gls@save@numberlist{##1}}%
1722   \fi
1723 }%
1724 }

```

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.

```

1725 \renewcommand*{\glsentryfmt}{%
1726   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}{}}%
1727   \glsifregular{\glslabel}%
1728   {\glstrregularfont{\glsgenentryfmt}}%
1729   {%
1730     \ifglshasshort{\glslabel}%
1731     {\glstrgenabbrvfmt}%
1732     {\glstrregularfont{\glsgenentryfmt}}%
1733   }%
1734 }

```

glstrregularfont Font used for regular entries.

```

1735 \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.

```

1736 \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).

```
1737 \@glstr@record{#2}{#3}{glslink}%
1738 \glsoifexists{#3}%
1739 {%
```

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).

```
1740 \let\glstrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
1741 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1742 \def\glscustomtext{#4}%
1743 \@glstr@field@linkdefs
1744 #1%
1745 \@gls@link[#2]{#3}{#4}%
1746 \let\ifKV@glslink@hyper\glstrorg@ifKV@glslink@hyper
1747 }%
1748 \glspostlinkhook
1749 }
```

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

\@gls@ Save the original definition and redefine.

```
1750 \let\@glstr@org@gls@\@gls@
1751 \def\@gls@#1#2{%
1752 \@glstr@record{#1}{#2}{glslink}%
1753 \@glstr@org@gls@{#1}{#2}%
1754 }%
```

\@glsp1@ Save the original definition and redefine.

```
1755 \let\@glstr@org@glsp1@\@glsp1@
1756 \def\@glsp1@#1#2{%
1757 \@glstr@record{#1}{#2}{glslink}%
1758 \@glstr@org@glsp1@{#1}{#2}%
1759 }%
```

\@Gls@ Save the original definition and redefine.

```
1760 \let\@glstr@org@Gls@\@Gls@
1761 \def\@Gls@#1#2{%
1762 \@glstr@record{#1}{#2}{glslink}%
1763 \@glstr@org@Gls@{#1}{#2}%
1764 }%
```

\@Glsp1@ Save the original definition and redefine.

```
1765 \let\@glstr@org@Glsp1@\@Glsp1@
1766 \def\@Glsp1@#1#2{%
1767 \@glstr@record{#1}{#2}{glslink}%
1768 \@glstr@org@Glsp1@{#1}{#2}%
1769 }%
```

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

```
1770 \let\@glxtr@org@GLS@\@GLS@
1771 \def\@GLS@#1#2{%
1772   \@glxtr@record{#1}{#2}{glslink}%
1773   \@glxtr@org@GLS@{#1}{#2}%
1774 }%
```

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

```
1775 \let\@glxtr@org@GLSpl@\@GLSpl@
1776 \def\@GLSpl@#1#2{%
1777   \@glxtr@record{#1}{#2}{glslink}%
1778   \@glxtr@org@GLSpl@{#1}{#2}%
1779 }%
```

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

```
1780 \renewcommand*{\@glsdisp}[3][{}]{%
1781   \@glxtr@record{#1}{#2}{glslink}%
1782   \glsdoifexists{#2}{%
1783     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
1784     \let\glsifplural\@secondoftwo
1785     \let\glscapscase\@firstofthree
1786     \def\glscustomtext{#3}%
1787     \def\glsinsert{}%
1788     \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
1789     \@gl@link[#1]{#2}{\@glo@text}%
1790     \ifKV@glslink@local
1791       \glslocalunset{#2}%
1792     \else
1793       \glsunset{#2}%
1794     \fi
1795   }%
1796   \glspostlinkhook
1797 }
```

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

```
1798 \renewcommand*{\@gls@link}[3][{}]{%
1799   \@glxtr@record{#1}{#2}{glslink}%
1800   \glsdoifexistsordo{#2}%
1801   {%
1802     \let\do@gl@link@checkfirsthyper\relax
1803     \@gl@link[#1]{#2}{#3}%
1804   }%
1805   {%
1806     \glstextformat{#3}%
1807   }%
1808   \glspostlinkhook
1809 }
```

gsxtrinitwrgloss Set the default if the wrgloss is omitted.

```
1810 \newcommand*{\gsxtrinitwrgloss}{%
1811   \glsifattribute{\glslabel}{wrgloss}{after}%
1812   {%
1813     \gsxtrinitwrglossbeforefalse
1814   }%
1815   {%
1816     \gsxtrinitwrglossbeforetrue
1817   }%
1818 }
```

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

```
1819 \newif\ifgsxtrinitwrglossbefore
1820 \gsxtrinitwrglossbeforetrue
```

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

```
1821 \define@choicekey{glslink}{wrgloss}%
1822 [\@gsxtr@wrglossval\@gsxtr@wrglossnr]%
1823 {before,after}%
1824 {%
1825   \ifcase\@gsxtr@wrglossnr\relax
1826     \gsxtrinitwrglossbeforetrue
1827   \or
1828     \gsxtrinitwrglossbeforefalse
1829   \fi
1830 }
```

```
1831 \define@key{glslink}{thevalue}{\def\@gsxtr@thevalue{#1}}
```

```
1832 \define@key{glslink}{theHvalue}{\def\@gsxtr@theHvalue{#1}}
```

gsxtr@hyperoutside Define a hyperoutside key to determine whether \hyperlink should be outside \glsformattext.

```
1833 \define@boolkey{glslink}[gsxtr@]{hyperoutside}[true]{}
1834 \gsxtr@hyperoutsidetrue
```

gsxtrinithyperoutside Set the default if the hyperoutside is omitted.

```
1835 \newcommand*{\gsxtrinithyperoutside}{%
1836   \glsifattribute{\glslabel}{hyperoutside}{false}%
1837   {%
1838     \gsxtr@hyperoutsidefalse
1839   }%
1840   {%
1841     \gsxtr@hyperoutsidetrue
1842   }%
1843 }
```

gsxtr@inc@linkcount Does nothing by default.

```
1844 \newcommand*{\gsxtr@inc@linkcount}{}%
```

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

```
1845 \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`.

```
1846 \def\@gls@link[#1]#2#3{%
1847   \leavevmode
1848   \edef\glslabel{\glsdetoklabel{#2}}%
1849   \def\@gls@link@opts{#1}%
1850   \let\@gls@link@label\glslabel
1851   \let\@glsnumberformat\@glsxtr@defaultnumberformat
1852   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
1853   \edef\glstype{\csname glo@\glslabel @type\endcsname}%
1854   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

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

```
1855   \def\@glsxtr@thevalue{%
1856   \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

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

```
1857   \glsxtrinitwrgloss
```

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

```
1858   \glsxtrinithyperoutside
```

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

```
1859   \@gls@setdefault@glslink@opts
```

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

```
1860   \glsxtr@inc@linkcount
```

As the original definition.

```
1861   \do@glsdisablehyperinlist
1862   \do@gls@link@checkfirsthyper
```

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

```
1863   \glslinkpresetkeys
```

Set options.

```
1864   \setkeys{glslink}{#1}%
```

User hook after options are set:

```
1865   \glslinkpostsetkeys
```

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

```
1866   \ifdefempty{\@glsxtr@thevalue}%
1867   {%
1868     \@gls@saveentrycounter
1869   }%
1870   {%
1871     \let\theglsentrycounter\@glsxtr@thevalue
1872     \def\theHglentrycounter{\@glsxtr@theHvalue}%
```

```

1873 }%
1874 \@gls@setsort{\glslabel}%
    Check textformat attribute (new to v1.21).
1875 \gls@hasattribute{\glslabel}{textformat}%
1876 {%
1877     \edef\@glsxtr@attrval{\gls@getattribute{\glslabel}{textformat}}%
1878     \ifcsdef{\@glsxtr@attrval}%
1879     {%
1880         \letcs{\@glsxtr@textformat}{\@glsxtr@attrval}%
1881     }%
1882 {%
1883     \GlossariesExtraWarning{Unknown control sequence name
1884     '\@glsxtr@attrval' supplied in textformat attribute
1885     for entry '\glslabel'. Reverting to default \string\glstextformat}%
1886     \let\@glsxtr@textformat\glstextformat
1887 }%
1888 }%
1889 {%
1890     \let\@glsxtr@textformat\glstextformat
1891 }%

```

Do write if it should occur before the link text:

```

1892 \ifglsxtrinitwrglossbefore
1893 \@do@wrglossary{#2}%
1894 \fi

```

Do the link text:

```

1895 \ifKV@glslink@hyper
1896 \ifglsxtr@hyperoutside
1897     \@glslink{\glslinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
1898 \else
1899     \@glsxtr@textformat{\@glslink{\glslinkprefix\glslabel}{#3}}%
1900 \fi
1901 \else
1902 \ifglsxtr@hyperoutside
1903     \glsdonohyperlink{\glslinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
1904 \else
1905     \@glsxtr@textformat{\glsdonohyperlink{\glslinkprefix\glslabel}{#3}}%
1906 \fi
1907 \fi

```

Do write if it should occur after the link text:

```

1908 \ifglsxtrinitwrglossbefore
1909 \else
1910     \@do@wrglossary{#2}%
1911 \fi

```

As the original definition:

```

1912 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
1913 }

```

```

1914 \define@key{glossadd}{thevalue}{\def\@glxtr@thevalue{#1}}
1915 \define@key{glossadd}{theHvalue}{\def\@glxtr@theHvalue{#1}}

```

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

```

1916 \renewrobustcmd*{\glsadd}[2][\%
1917   \glxtrifinmark
1918   }{%
1919   {%
1920     \@gls@adjustmode
1921     \@glxtr@record{#1}{#2}{glossadd}%
1922     \glsdoifexists{#2}%
1923     {%
1924       \let\@glsnumberformat\@glxtr@defaultnumberformat
1925       \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
1926       \def\@glxtr@thevalue{%
1927         \def\@glxtr@theHvalue{\@glxtr@thevalue}%
1928         \setkeys{glossadd}{#1}%
1929         \ifdefempty{\@glxtr@thevalue}%
1930         {%
1931           \@gls@saveentrycounter
1932         }%
1933         {%
1934           \let\theglentrycounter\@glxtr@thevalue
1935           \def\theHglentrycounter{\@glxtr@theHvalue}%
1936         }%

```

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

```

1937       \@gls@setsort{#2}%
1938       \@do@wrglossary{#2}%
1939     }%
1940   }%
1941 }

```

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

```

1942 \newcommand*{\@glxtr@field@linkdefs}{%
1943   \let\glxtrifwasfirstuse\@secondoftwo
1944   \let\glsifplural\@secondoftwo
1945   \let\glscapscase\@firstofthree
1946   \let\glsinsert\@empty
1947 }

```

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

`assignfieldfont`

```

1948 \newcommand*{\glxtrassignfieldfont}[1]{%
1949   \ifglentryexists{#1}%
1950   {%
1951     \ifglshasshort{#1}%

```

```

1952   {%
1953     \glssetabbrvfmt{\glscategory{#1}}%
1954     \glsifregular{#1}%
1955     {\let\@gls@field@font\glstrregularfont}%
1956     {\let\@gls@field@font\@firstofone}%
1957   }%
1958   {%
1959     \glsifnotregular{#1}%
1960     {\let\@gls@field@font\@firstofone}%
1961     {\let\@gls@field@font\glstrregularfont}%
1962   }%
1963 }%
1964 {%
1965   \let\@gls@field@font\@gobble
1966 }%
1967 }

```

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

```

1968 \def\@glstext@#1#2[#3]{%
1969   \glstrassignfieldfont{#2}%
1970   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
1971 }

```

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

```

1972 \def\@GLStext@#1#2[#3]{%
1973   \glstrassignfieldfont{#2}%
1974   \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
1975   {\@gls@field@font{\@GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
1976 }

```

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

```

1977 \def\@Glstext@#1#2[#3]{%
1978   \glstrassignfieldfont{#2}%
1979   \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
1980   {\@gls@field@font{\@GLSaccesstext{#2}#3}}%
1981 }

```

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`

```

1982 \newcommand*{\glstrchecknohyperfirst}[1]{%
1983   \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
1984 }

```

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

```

1985 \def\@glsfirst@#1#2[#3]{%
1986   \glstrassignfieldfont{#2}%

```

Ensure that \glsfirst honours the nohyperfirst attribute.

```
1987 \@gls@field@link
1988 [\let\glstrifwasfirstuse\@firstoftwo
1989 \glstrchecknohyperfirst{#2}%
1990 ]{#1}{#2}%
1991 {\@gls@field@font{\glsaccessfirst{#2}#3}}%
1992 }
```

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

```
1993 \def\@Glsfirst@#1#2[#3]{%
1994 \glstrassignfieldfont{#2}%
1995 \@gls@field@link
1996 [\let\glstrifwasfirstuse\@firstoftwo
1997 \let\glscapscase\@secondofthree
1998 \glstrchecknohyperfirst{#2}%
1999 ]%
2000 {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
2001 }
```

Ensure that \Glsfirst honours the nohyperfirst attribute.

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

```
2002 \def\@GLSfirst@#1#2[#3]{%
2003 \glstrassignfieldfont{#2}%
2004 \@gls@field@link
2005 [\let\glstrifwasfirstuse\@firstoftwo
2006 \let\glscapscase\@thirdofthree
2007 \glstrchecknohyperfirst{#2}%
2008 ]%
2009 {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2010 }
```

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

```
2011 \def\@glsplural@#1#2[#3]{%
2012 \glstrassignfieldfont{#2}%
2013 \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
2014 {\@gls@field@font{\glsaccessplural{#2}#3}}%
2015 }
```

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

```
2016 \def\@Glsplural@#1#2[#3]{%
2017 \glstrassignfieldfont{#2}%
2018 \@gls@field@link
2019 [\let\glsifplural\@firstoftwo
2020 \let\glscapscase\@secondofthree
2021 ]%
2022 {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
2023 }
```

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

```

2024 \def\@GLSplural@#1#2[#3]{%
2025   \glstrassignfieldfont{#2}%
2026   \@gls@field@link
2027   [\let\glsifplural\@firstoftwo
2028   \let\glscapscase\@thirdofthree
2029   ]%
2030   {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2031 }

```

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

```

2032 \def\@glsfirstplural@#1#2[#3]{%
2033   \glstrassignfieldfont{#2}%

```

Ensure that \glsfirstplural honours the nohyperfirst attribute.

```

2034   \@gls@field@link
2035   [\let\glstrifwasfirstuse\@firstoftwo
2036   \let\glsifplural\@firstoftwo
2037   \glstrchecknohyperfirst{#2}%
2038   ]%
2039   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
2040 }

```

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

```

2041 \def\@Glsfirstplural@#1#2[#3]{%
2042   \glstrassignfieldfont{#2}%

```

Ensure that \glsfirstplural honours the nohyperfirst attribute.

```

2043   \@gls@field@link
2044   [\let\glstrifwasfirstuse\@firstoftwo
2045   \let\glsifplural\@firstoftwo
2046   \let\glscapscase\@secondofthree
2047   \glstrchecknohyperfirst{#2}%
2048   ]%
2049   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
2050 }

```

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

```

2051 \def\@GLSfirstplural@#1#2[#3]{%
2052   \glstrassignfieldfont{#2}%

```

Ensure that \glsfirstplural honours the nohyperfirst attribute.

```

2053   \@gls@field@link
2054   [\let\glstrifwasfirstuse\@firstoftwo
2055   \let\glsifplural\@firstoftwo
2056   \let\glscapscase\@thirdofthree
2057   \glstrchecknohyperfirst{#2}%
2058   ]%
2059   {#1}{#2}%
2060   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2061 }

```

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

```
2062 \def\@glsname@#1#2[#3]{%
2063   \glstrassignfieldfont{#2}%
2064   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessname{#2}#3}}%
2065 }
```

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

```
2066 \def\@Glsname@#1#2[#3]{%
2067   \glstrassignfieldfont{#2}%
2068   \@gls@field@link
2069   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2070   {\@gls@field@font{\@Glsaccessname{#2}#3}}%
2071 }
```

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

```
2072 \def\@GLSname@#1#2[#3]{%
2073   \glstrassignfieldfont{#2}%
2074   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2075   {#1}{#2}%
2076   {\@gls@field@font{\@GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2077 }
```

\@glsdesc@

```
2078 \def\@glsdesc@#1#2[#3]{%
2079   \glstrassignfieldfont{#2}%
2080   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessdesc{#2}#3}}%
2081 }
```

\@Glsdesc@ First letter uppercase version.

```
2082 \def\@Glsdesc@#1#2[#3]{%
2083   \glstrassignfieldfont{#2}%
2084   \@gls@field@link
2085   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2086   {\@gls@field@font{\@Glsaccessdesc{#2}#3}}%
2087 }
```

\@GLSdesc@ All uppercase version.

```
2088 \def\@GLSdesc@#1#2[#3]{%
2089   \glstrassignfieldfont{#2}%
2090   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2091   {#1}{#2}{\@gls@field@font{\@GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2092 }
```

@glsdescplural@ No case-changing version.

```
2093 \def\@glsdescplural@#1#2[#3]{%
2094   \glstrassignfieldfont{#2}%
2095   \@gls@field@link
2096   [\let\glscapscase\@secondoftwo
```

```

2097 \let\glsifplural\@firstoftwo
2098 ]{\#1}{\#2}{\@gls@field@font{\glsaccessdescplural{\#2}{\#3}}}%
2099 }

```

@Glsdescplural@ First letter uppercase version.

```

2100 \def\@Glsdescplural@#1#2[#3]{%
2101 \glsxtrassignfieldfont{\#2}%
2102 \@gls@field@link
2103 [\let\glscapscase\@secondoftwo
2104 \let\glsifplural\@firstoftwo
2105 ]{\#1}{\#2}{\@gls@field@font{\Glsaccessdescplural{\#2}{\#3}}}%
2106 }

```

@GLSdescplural@ All uppercase version.

```

2107 \def\@GLSdesc@#1#2[#3]{%
2108 \glsxtrassignfieldfont{\#2}%
2109 \@gls@field@link
2110 [\let\glscapscase\@thirdoftwo
2111 \let\glsifplural\@firstoftwo
2112 ]%
2113 {\#1}{\#2}%
2114 {\@gls@field@font{\GLSaccessdescplural{\#2}\mfirstucMakeUppercase{\#3}}}%
2115 }

```

\@glssymbol@

```

2116 \def\@glssymbol@#1#2[#3]{%
2117 \glsxtrassignfieldfont{\#2}%
2118 \@gls@field@link{\#1}{\#2}{\@gls@field@font{\glsaccesssymbol{\#2}{\#3}}}%
2119 }

```

\@Glsymbol@ First letter uppercase version.

```

2120 \def\@Glsymbol@#1#2[#3]{%
2121 \glsxtrassignfieldfont{\#2}%
2122 \@gls@field@link
2123 [\let\glscapscase\@secondoftwo]%
2124 {\#1}{\#2}{\@gls@field@font{\Glsaccesssymbol{\#2}{\#3}}}%
2125 }

```

\@GLSsymbol@ All uppercase version.

```

2126 \def\@GLSsymbol@#1#2[#3]{%
2127 \glsxtrassignfieldfont{\#2}%
2128 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2129 {\#1}{\#2}{\@gls@field@font{\GLSaccesssymbol{\#2}\mfirstucMakeUppercase{\#3}}}%
2130 }

```

lssymbolplural@ No case-changing version.

```

2131 \def\@glssymbolplural@#1#2[#3]{%
2132 \glsxtrassignfieldfont{\#2}%

```

```

2133 \@gls@field@link
2134 [\let\glscapscase\@secondoftwo
2135 \let\glsifplural\@firstoftwo
2136 ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2137 }

```

lssymbolplural@ First letter uppercase version.

```

2138 \def\@GLssymbolplural@#1#2[#3]{%
2139 \glstrassignfieldfont{#2}%
2140 \@gls@field@link
2141 [\let\glscapscase\@secondoftwo
2142 \let\glsifplural\@firstoftwo
2143 ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
2144 }

```

LSsymbolplural@ All uppercase version.

```

2145 \def\@GLSsymbol@#1#2[#3]{%
2146 \glstrassignfieldfont{#2}%
2147 \@gls@field@link
2148 [\let\glscapscase\@thirdoftwo
2149 \let\glsifplural\@firstoftwo
2150 ]%
2151 {#1}{#2}%
2152 {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2153 }

```

\@Glsuseri@ First letter uppercase version.

```

2154 \def\@Glsuseri@#1#2[#3]{%
2155 \glstrassignfieldfont{#2}%
2156 \@gls@field@link
2157 [\let\glscapscase\@secondoftwo]{#1}{#2}%
2158 {\@gls@field@font{\Glsentryuseri{#2}#3}}%
2159 }

```

\@GLSuseri@ All uppercase version.

```

2160 \def\@GLSuseri@#1#2[#3]{%
2161 \glstrassignfieldfont{#2}%
2162 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2163 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}}%
2164 }

```

\@Glsuserii@ First letter uppercase version.

```

2165 \def\@Glsuserii@#1#2[#3]{%
2166 \glstrassignfieldfont{#2}%
2167 \@gls@field@link
2168 [\let\glscapscase\@secondoftwo]%
2169 {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
2170 }

```

\@GLSuserii@ All uppercase version.

```
2171 \def\@GLSuserii@#1#2[#3]{%
2172   \glstrassignfieldfont{#2}%
2173   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2174   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2175 }
```

\@Glsuseriii@ First letter uppercase version.

```
2176 \def\@Glsuseriii@#1#2[#3]{%
2177   \glstrassignfieldfont{#2}%
2178   \@gls@field@link
2179   [\let\glscapscase\@secondoftwo]%
2180   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}}%
2181 }
```

\@GLSuseriii@ All uppercase version.

```
2182 \def\@GLSuseriii@#1#2[#3]{%
2183   \glstrassignfieldfont{#2}%
2184   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2185   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2186 }
```

\@Glsuseriv@ First letter uppercase version.

```
2187 \def\@Glsuseriv@#1#2[#3]{%
2188   \glstrassignfieldfont{#2}%
2189   \@gls@field@link
2190   [\let\glscapscase\@secondoftwo]%
2191   {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}}%
2192 }
```

\@GLSuseriv@ All uppercase version.

```
2193 \def\@GLSuseriv@#1#2[#3]{%
2194   \glstrassignfieldfont{#2}%
2195   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2196   {#1}{#2}%
2197   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2198 }
```

\@Glsuserv@ First letter uppercase version.

```
2199 \def\@Glsuserv@#1#2[#3]{%
2200   \glstrassignfieldfont{#2}%
2201   \@gls@field@link
2202   [\let\glscapscase\@secondoftwo]%
2203   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}}%
2204 }
```

\@GLSuserv@ All uppercase version.

```
2205 \def\@GLSuserv@#1#2[#3]{%
```

```

2206 \glstrassignfieldfont{#2}%
2207 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2208 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2209 }

```

\@Glsuservi@ First letter uppercase version.

```

2210 \def\@Glsuservi@#1#2[#3]{%
2211 \glstrassignfieldfont{#2}%
2212 \@gls@field@link
2213 [\let\glscapscase\@secondoftwo]%
2214 {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}}%
2215 }

```

\@GLSuservi@ All uppercase version.

```

2216 \def\@GLSuservi@#1#2[#3]{%
2217 \glstrassignfieldfont{#2}%
2218 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2219 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
2220 }

```

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

\@acrshort No case change.

```

2221 \def\@acrshort#1#2[#3]{%
2222 \glsdoifexists{#2}%
2223 {%
2224 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2225 \let\glxtrifwasfirstuse\@secondoftwo
2226 \let\glsifplural\@secondoftwo
2227 \let\glscapscase\@firstofthree
2228 \let\glsinsert\@empty
2229 \def\glscustomtext{%
2230 \acronymfont{\glsaccesssshort{#2}}#3%
2231 }%
2232 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2233 }%
2234 \glspostlinkhook
2235 }

```

\@Acrshort First letter uppercase.

```

2236 \def\@Acrshort#1#2[#3]{%
2237 \glsdoifexists{#2}%
2238 {%
2239 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2240 \let\glxtrifwasfirstuse\@secondoftwo
2241 \let\glsifplural\@secondoftwo
2242 \let\glscapscase\@secondofthree
2243 \let\glsinsert\@empty

```

```

2244 \def\glscustomtext{%
2245 \acronymfont{\Glsaccesssshort{#2}}#3%
2246 }%
2247 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2248 }%
2249 \glspostlinkhook
2250 }

```

\@ACRshort All uppercase.

```

2251 \def\@ACRshort#1#2[#3]{%
2252 \glsdofexists{#2}%
2253 {%
2254 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2255 \let\glstrifwasfirstuse\@secondoftwo
2256 \let\glsifplural\@secondoftwo
2257 \let\glscapscase\@thirdofthree
2258 \let\glsinsert\@empty
2259 \def\glscustomtext{%
2260 \mfirstucMakeUppercase{\acronymfont{\Glsaccesssshort{#2}}#3}%
2261 }%
2262 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2263 }%
2264 \glspostlinkhook
2265 }

```

\@acrshortpl No case change.

```

2266 \def\@acrshortpl#1#2[#3]{%
2267 \glsdofexists{#2}%
2268 {%
2269 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2270 \let\glstrifwasfirstuse\@secondoftwo
2271 \let\glsifplural\@firstoftwo
2272 \let\glscapscase\@firstofthree
2273 \let\glsinsert\@empty
2274 \def\glscustomtext{%
2275 \acronymfont{\Glsaccesssshortpl{#2}}#3%
2276 }%
2277 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2278 }%
2279 \glspostlinkhook
2280 }

```

\@Acrshortpl First letter uppercase.

```

2281 \def\@Acrshortpl#1#2[#3]{%
2282 \glsdofexists{#2}%
2283 {%
2284 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2285 \let\glstrifwasfirstuse\@secondoftwo
2286 \let\glsifplural\@firstoftwo

```

```

2287 \let\glscapscase\@secondofthree
2288 \let\glsinsert\@empty
2289 \def\glscustomtext{%
2290 \acronymfont{\Glsaccessshortpl{#2}}#3%
2291 }%
2292 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2293 }%
2294 \glspostlinkhook
2295 }

```

\@ACRshortpl All uppercase.

```

2296 \def\@ACRshortpl#1#2[#3]{%
2297 \glsdoifexists{#2}%
2298 {%
2299 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2300 \let\glxtrifwasfirstuse\@secondoftwo
2301 \let\glsifplural\@firstoftwo
2302 \let\glscapscase\@thirdofthree
2303 \let\glsinsert\@empty
2304 \def\glscustomtext{%
2305 \mfirstucMakeUppercase{\acronymfont{\Glsaccessshortpl{#2}}#3}%
2306 }%
2307 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2308 }%
2309 \glspostlinkhook
2310 }

```

\@acrlong No case change.

```

2311 \def\@acrlong#1#2[#3]{%
2312 \glsdoifexists{#2}%
2313 {%
2314 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2315 \let\glxtrifwasfirstuse\@secondoftwo
2316 \let\glsifplural\@secondoftwo
2317 \let\glscapscase\@firstofthree
2318 \let\glsinsert\@empty
2319 \def\glscustomtext{%
2320 \acronymfont{\Glsaccesslong{#2}}#3%
2321 }%
2322 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2323 }%
2324 \glspostlinkhook
2325 }

```

\@Acrlong First letter uppercase.

```

2326 \def\@Acrlong#1#2[#3]{%
2327 \glsdoifexists{#2}%
2328 {%
2329 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

2330 \let\glxtrifwasfirstuse\@secondoftwo
2331 \let\glsifplural\@secondoftwo
2332 \let\glscapscase\@secondofthree
2333 \let\glsinsert\@empty
2334 \def\glscustomtext{%
2335     \acronymfont{\Glsaccesslong{#2}}#3%
2336 }%
2337 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2338 }%
2339 \glspostlinkhook
2340 }

```

\@ACRlong All uppercase.

```

2341 \def\@ACRlong#1#2[#3]{%
2342     \glsdoifexists{#2}%
2343     {%
2344         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2345         \let\glxtrifwasfirstuse\@secondoftwo
2346         \let\glsifplural\@secondoftwo
2347         \let\glscapscase\@thirdofthree
2348         \let\glsinsert\@empty
2349         \def\glscustomtext{%
2350             \mfirstucMakeUppercase{\acronymfont{\Glsaccesslong{#2}}#3}%
2351         }%
2352         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2353     }%
2354     \glspostlinkhook
2355 }

```

\@acrlongpl No case change.

```

2356 \def\@acrlongpl#1#2[#3]{%
2357     \glsdoifexists{#2}%
2358     {%
2359         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2360         \let\glxtrifwasfirstuse\@secondoftwo
2361         \let\glsifplural\@firstoftwo
2362         \let\glscapscase\@firstofthree
2363         \let\glsinsert\@empty
2364         \def\glscustomtext{%
2365             \acronymfont{\Glsaccesslongpl{#2}}#3%
2366         }%
2367         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2368     }%
2369     \glspostlinkhook
2370 }

```

\@Acrlongpl First letter uppercase.

```

2371 \def\@Acrlongpl#1#2[#3]{%
2372     \glsdoifexists{#2}%

```

```

2373 {%
2374   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2375   \let\glxtrifwasfirstuse\@secondoftwo
2376   \let\gl@sifplural\@firstoftwo
2377   \let\glscapscase\@secondofthree
2378   \let\gl$insert\@empty
2379   \def\glscustomtext{%
2380     \acronymfont{\Glsaccesslongpl{#2}}#3%
2381   }%
2382   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2383 }%
2384 \glspostlinkhook
2385 }

```

\@ACRlongpl All uppercase.

```

2386 \def\@ACRlongpl#1#2[#3]{%
2387   \glsdofexists{#2}%
2388   {%
2389     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2390     \let\glxtrifwasfirstuse\@secondoftwo
2391     \let\gl@sifplural\@firstoftwo
2392     \let\glscapscase\@thirdofthree
2393     \let\gl$insert\@empty
2394     \def\glscustomtext{%
2395       \mfirstucMakeUppercase{\acronymfont{\Glsaccesslongpl{#2}}#3}%
2396     }%
2397     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2398   }%
2399   \glspostlinkhook
2400 }

```

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

\@glsaddkey

```

2401 \renewcommand*{\@glsaddkey}[7]{%
2402   \key@ifundefined{glossentry}{#1}%
2403   {%
2404     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2405     \appto\@gl@keymap{, {#1}{#1}}%
2406     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
2407     \appto\@newglossaryentryposthook{%
2408       \letcs{\@glo@tmp}{@glo@#1}%
2409       \gl@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2410     }%
2411     \newcommand*{#3}[1]{\@gl@entry@field{##1}{#1}}%
2412     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

2413   \ifcsdef{@gl@user@#1@}%

```

```

2414 {%
2415   \PackageError{glossaries}%
2416   {Can't define '\string#5' as helper command
2417   '\expandafter\string\csname @gls@user@#1@\endcsname' already
2418   exists}%
2419   {}}%
2420 }%
2421 {%
2422   \expandafter\newcommand\expandafter*\expandafter
2423   {\csname @gls@user@#1\endcsname}[2][{}]{%
2424     \new@ifnextchar[%
2425       {\csuse{@gls@user@#1@}{##1}{##2}}%
2426       {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
2427   \csdef{@gls@user@#1@}##1##2[##3]{%
2428     \@gls@field@link{##1}{##2}{#3{##2}##3}%
2429   }%
2430   \newrobustcmd*{#5}{%
2431     \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2432   }%

```

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

```

2433   \ifcsdef{@Gls@user@#1@}%
2434   {%
2435     \PackageError{glossaries}%
2436     {Can't define '\string#6' as helper command
2437     '\expandafter\string\csname @Gls@user@#1@\endcsname' already
2438     exists}%
2439     {}}%
2440   }%
2441   {%
2442     \expandafter\newcommand\expandafter*\expandafter
2443     {\csname @Gls@user@#1\endcsname}[2][{}]{%
2444       \new@ifnextchar[%
2445         {\csuse{@Gls@user@#1@}{##1}{##2}}%
2446         {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
2447     \csdef{@Gls@user@#1@}##1##2[##3]{%
2448       \@gls@field@link[\let\glscaps@case\@secondofthree]%
2449       {##1}{##2}{#4{##2}##3}%
2450     }%
2451     \newrobustcmd*{#6}{%
2452       \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2453     }%

```

Finally the all caps version (modified):

```

2454   \ifcsdef{@GLS@user@#1@}%
2455   {%
2456     \PackageError{glossaries}%
2457     {Can't define '\string#7' as helper command
2458     '\expandafter\string\csname @GLS@user@#1@\endcsname' already
2459     exists}%

```

```

2460     {}%
2461 }%
2462 {%
2463     \expandafter\newcommand\expandafter*\expandafter
2464     {\csname @GLS@user@#1\endcsname}[2][{}]{%
2465         \new@ifnextchar[%
2466             {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2467             {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
2468     \csdef{@GLS@user@#1@}##1##2[##3]{%
2469         \@gls@field@link[\let\gls@caps@case\@thirdofthree]%
2470         {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2471     }%
2472     \newrobustcmd*{#7}{%
2473         \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2474     }%
2475 }%
2476 {%
2477     \PackageError{glossaries-extra}{Key ‘#1’ already exists}{}%
2478 }%
2479 }

```

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

```

2480 \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.

```

2481 \let\@gls@xtr@org@checkfirsthyper\@gls@link@checkfirsthyper
2482 \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.

```

2483 \ifglsused{\glslabel}%
2484 {\let\gls@xtrifwasfirstuse\@secondoftwo}
2485 {\let\gls@xtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```

2486 \edef\gls@categorylabel{\gls@category{\glslabel}}%
2487 \ifglsused{\glslabel}%
2488 {%
2489     \glsifcategoryattribute{\gls@categorylabel}{nohypernext}{true}%
2490     {\KV@glslink@hyperfalse}}}%
2491 }%
2492 {%
2493     \glsifcategoryattribute{\gls@categorylabel}{nohyperfirst}{true}%
2494     {\KV@glslink@hyperfalse}}}%
2495 }%
2496 \glslinkcheckfirsthyperhook
2497 }

```

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.

```
2498 \ifdef\do@glsglssdisablehyperinlist
2499 {%
2500   \let\@glsglssxtr@do@glsglssdisablehyperinlist\do@glsglssdisablehyperinlist
2501   \renewcommand*{\do@glsglssdisablehyperinlist}{%
2502     \@glsglssxtr@do@glsglssdisablehyperinlist
2503     \glsglssifattribute{\glsglsslabel}{nohyper}{true}{\KV@glsglsslink@hyperfalse}}}%
2504   }
2505 }
2506 {}
```

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

```
2507 \define@boolkey{glsglsslink}{noindex}[true]{}
2508 \KV@glsglsslink@noindexfalse
```

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

lt@glsglsslink@opts

```
2509 \ifdef\@glsglss@setdefault@glsglsslink@opts
2510 {
2511   \renewcommand*{\@glsglss@setdefault@glsglsslink@opts}{%
2512     \KV@glsglsslink@noindexfalse
2513     \@glsglssxtrsetaliasnoindex
2514   }
2515 }
2516 {
```

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

```
2517   \newcommand*{\@glsglss@setdefault@glsglsslink@opts}{%
2518     \KV@glsglsslink@noindexfalse
2519     \@glsglssxtrsetaliasnoindex
2520   }
2521   \preto\do@glsglssdisablehyperinlist{\@glsglss@setdefault@glsglsslink@opts}
2522 }
```

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.)

```
2523 \providecommand*{\glsglssxtrsetaliasnoindex}{%
2524   \KV@glsglsslink@noindextrue
2525 }
```

setaliasnoindex

```
2526 \newcommand*{\@glsglssxtrsetaliasnoindex}{%
2527   \glsglssxtrifhasfield{alias}{\glsglsslabel}%
2528   {%
2529     \let\glsglssxtrindexaliased\@glsglssxtrindexaliased
```

```

2530 \glxtrsetaliasnoindex
2531 \let\glxtrindexaliased\@no@glxtrindexaliased
2532 }%
2533 {}%
2534 }

```

xtrindexaliased

```

2535 \newcommand{\@glxtrindexaliased}{%
2536 \ifKV@glslink@noindex
2537 \else
2538 \begingroup
2539 \let\@glslnumberformat\@glxtr@defaultnumberformat
2540 \edef\@glslcounter{\csname glo@\glsl@detoklabel{\glsl@label}\@counter\endcsname}%
2541 \glxtr@saveentrycounter
2542 \@do@wrglossary{\glxtr@alias{\glsl@label}}%
2543 \endgroup
2544 \fi
2545 }

```

xtrindexaliased

```

2546 \newcommand{\@no@glxtrindexaliased}{%
2547 \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
2548 not permitted outside definition of \string\glxtrsetaliasnoindex}%
2549 {}%
2550 }

```

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

```

2551 \let\glxtrindexaliased\@no@glxtrindexaliased

```

tDefaultGlsOpts Set the default options for \glslink etc.

```

2552 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
2553 \renewcommand*{\@glsl@setdefault@glslink@opts}{%
2554 \setkeys{glslink}{#1}%
2555 \@glxtrsetaliasnoindex
2556 }%
2557 }

```

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

```

2558 \newcommand*{\glxtrifindexing}[2]{%
2559 \ifKV@glslink@noindex #2\else #1\fi
2560 }

```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```

2561 \renewcommand*{\glswriteentry}[2]{%
2562 \glxtrifindexing
2563 {%
2564 \ifglslindexonlyfirst
2565 \ifglslused{#1}

```

```

2566     {\glxtrdoautoindexname{#1}{dualindex}}}%
2567     {#2}}%
2568   \else
2569     \glusifattribute{#1}{indexonlyfirst}{true}%
2570     {\ifglused{#1}
2571      {\glxtrdoautoindexname{#1}{dualindex}}}%
2572      {#2}}}%
2573     {#2}}%
2574   \fi
2575 }%
2576 {}%
2577 }

```

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

```

2578 \appto\@do@@wrglossary{\@glxtr@do@@wrindex
2579   \glxtrdowrglossaryhook{\@gls@label}}%
2580 }

```

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

Similarly for the “noidx” version:

`s@noidxglossary`

```

2581 \appto\gls@noidxglossary{\@glxtr@do@@wrindex
2582   \glxtrdowrglossaryhook{\@gls@label}}%
2583 }

```

`xtr@do@@wrindex`

```

2584 \newcommand*{\@glxtr@do@@wrindex}{%
2585   \glxtrdoautoindexname{\@gls@label}{dualindex}}%
2586 }

```

`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.)

```

2587 \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.

```

2588 \newcommand*{\@gls@alt@hyp@opt}[1]{%
2589   \let\glslinkvar\@firstofthree
2590   \let\@gls@hyp@opt@cs#1\relax
2591   \@ifstar{\s@gls@hyp@opt}%
2592   {\@ifnextchar+%
2593     {\@firstoftwo{\p@gls@hyp@opt}}%
2594     {%
2595       \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
2596       {\@firstoftwo{\@alt@gls@hyp@opt}}%
2597       {#1}}%

```

```

2598 }%
2599 }%
2600 }

```

`\alt@glshyp@opt` User version

```

2601 \newcommand*{\@alt@glshyp@opt}[1][]{%
2602 \let\glslinkvar\@firstofthree
2603 \expandafter\@glshyp@opt@cs\expandafter[\@glshyp@opt@keys,#1]}

```

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

```

2604 \newcommand*{\@glshyp@opt@char}{}

```

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

```

2605 \newcommand*{\@glshyp@opt@keys}{}

```

`\rSetAltModifier`

```

2606 \newcommand*{\GlsXtrSetAltModifier}[2]{%
2607 \let\@glshyp@opt\@glshyp@opt
2608 \def\@glshyp@opt@char{#1}%
2609 \def\@glshyp@opt@keys{#2}%
2610 }

```

`\org@dohyperlink`

```

2611 \let\glsxtr@org@dohyperlink\glsdohyperlink

```

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

```

2612 \ifdef\glshyp@opt@char
2613 {
2614 \renewcommand*{\glshyp@opt@char}[3][\@glo@type]{%
2615 \edef\glshyp@opt@label{#2}\protected@edef\glshyp@opt@title{#3}%

```

Scope:

```

2616 {%
2617 \let\glsdohyperlink\glsxtr@org@dohyperlink
2618 \let\glslink\glshyp@opt@char{#1}{#2}{#3}%
2619 }%
2620 }%
2621 }
2622 {}

```

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

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

```

2623 \renewcommand*{\glsdohyperlink}[2]{%
2624   \glshasattribute{\glslabel}{targeturl}%
2625   {%
2626     \glshasattribute{\glslabel}{targetname}%
2627     {%
2628       \glshasattribute{\glslabel}{targetcategory}%
2629       {%
2630         \hyperref{\glsggetattribute{\glslabel}{targeturl}}{%
2631           {\glsggetattribute{\glslabel}{targetcategory}}}%
2632         {\glsggetattribute{\glslabel}{targetname}}}%
2633         {\glxtrprotectlinks#2}}%
2634       }%
2635     {%
2636       \hyperref{\glsggetattribute{\glslabel}{targeturl}}{%
2637         }%
2638         {\glsggetattribute{\glslabel}{targetname}}}%
2639         {\glxtrprotectlinks#2}}%
2640     }%
2641   }%
2642   {%
2643     \href{\glsggetattribute{\glslabel}{targeturl}}{%
2644       {\glxtrprotectlinks#2}}%
2645     }%
2646   }%
2647   {%

```

Check for alias.

```

2648   \glsgfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
2649   \ifdefvoid\gloaliaslabel
2650   {%
2651     \glxtrhyperlink{#1}{\glxtrprotectlinks#2}}%
2652   }%
2653   {%

```

Redirect link to the alias target.

```

2654     \glxtrhyperlink
2655     {\glolinkprefix\glsgdetoklabel{\gloaliaslabel}}%
2656     {\glxtrprotectlinks#2}}%
2657   }%
2658 }%
2659 }

```

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

```

2660 \ifdef\@glsshowtarget
2661 {
2662   \newcommand{\glxtrhyperlink}[2]{%
2663     \@glsshowtarget{#1}%
2664     \hyperlink{#1}{#2}%

```

```

2665 }%
2666 }
2667 {
2668   \newcommand{\glstrhyperlink}[2]{\hyperlink{#1}{#2}}%
2669 }

```

`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.

```

2670 \renewrobustcmd*{\glshyperlink}[2][\glstrytext{\@glo@label}]{%
2671   \glsdoifexists{#2}%
2672   {%
2673     \def\@glo@label{#2}%
2674     {\edef\glslabel{#2}%
2675       \@glslink{\glo@linkprefix\glslabel}{#1}}%
2676   }%
2677 }

```

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

```

2678 \renewcommand{\glsdisablehyper}{%
2679   \KV@glslink@hyperfalse
2680   \def\@glslink{\glsdohyperlink}%
2681   \let\@glstarget\@secondoftwo
2682 }

```

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

```

2683 \renewcommand{\glsenablehyper}{%
2684   \KV@glslink@hypertrue
2685   \def\@glslink{\glsdohyperlink}%
2686   \def\@glstarget{\glsdohypertarget}%
2687 }

```

`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.

```

2688 \def\glsdohyperlink#1#2{{\glstrprotectlinks #2}}

```

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

```

2689 \ifcsundef{hyperlink}%
2690 {%
2691   \def\@glslink{\glsdohyperlink}
2692 }%
2693 {%
2694   \def\@glslink{\glsdohyperlink}
2695 }

```

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

```

2696 \newcommand*{\glsxtrprotectlinks}{%
2697   \KV@glslink@hyperfalse
2698   \KV@glslink@noindextrue
2699   \let\@gls@\@glsxtr@p@text@
2700   \let\@Gls@\@Glsxtr@p@text@
2701   \let\@GLS@\@GLSxtr@p@text@
2702   \let\@glspl@\@glsxtr@p@plural@
2703   \let\@Glspl@\@Glsxtr@p@plural@
2704   \let\@GLSpl@\@GLSxtr@p@plural@
2705   \let\@glsxtrshort@\@glsxtr@p@short@
2706   \let\@Glsxtrshort@\@Glsxtr@p@short@
2707   \let\@GLSxtrshort@\@GLSxtr@p@short@
2708   \let\@glsxtrlong@\@glsxtr@p@long@
2709   \let\@Glsxtrlong@\@Glsxtr@p@long@
2710   \let\@GLSxtrlong@\@GLSxtr@p@long@
2711   \let\@glsxtrshortpl@\@glsxtr@p@shortpl@
2712   \let\@Glsxtrshortpl@\@Glsxtr@p@shortpl@
2713   \let\@GLSxtrshortpl@\@GLSxtr@p@shortpl@
2714   \let\@glsxtrlongpl@\@glsxtr@p@longpl@
2715   \let\@Glsxtrlongpl@\@Glsxtr@p@longpl@
2716   \let\@GLSxtrlongpl@\@GLSxtr@p@longpl@
2717   \let\@acrshort@\@glsxtr@p@acrshort@
2718   \let\@Acrshort@\@Glsxtr@p@acrshort@
2719   \let\@ACRshort@\@GLSxtr@p@acrshort@
2720   \let\@acrshortpl@\@glsxtr@p@acrshortpl@
2721   \let\@Acrshortpl@\@Glsxtr@p@acrshortpl@
2722   \let\@ACRshortpl@\@GLSxtr@p@acrshortpl@
2723   \let\@acrlong@\@glsxtr@p@acrlong@
2724   \let\@Acrlong@\@Glsxtr@p@acrlong@
2725   \let\@ACRlong@\@GLSxtr@p@acrlong@
2726   \let\@acrlongpl@\@glsxtr@p@acrlongpl@
2727   \let\@Acrlongpl@\@Glsxtr@p@acrlongpl@
2728   \let\@ACRlongpl@\@GLSxtr@p@acrlongpl@
2729 }

```

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

@glsxtr@p@text@

```

2730 \def\@glsxtr@p@text@#1#2[#3]{\@gls{text@{#1}{#2}[#3]}}

```

@Glsxtr@p@text@

```

2731 \def\@Glsxtr@p@text@#1#2[#3]{\@Gls{text@{#1}{#2}[#3]}}

```

@GLSxtr@p@text@

```

2732 \def\@GLSxtr@p@text@#1#2[#3]{\@GLS{text@{#1}{#2}[#3]}}

```

lsxtr@p@plural@

```

2733 \def\@glsxtr@p@plural@#1#2[#3]{\@gls{plural@{#1}{#2}[#3]}}

```

lsxtr@p@plural@

```
2734 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}
```

LSxtr@p@plural@

```
2735 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}
```

glxtr@p@short@

```
2736 \def\@glxtr@p@short@#1#2[#3]{%
2737   {%
2738     \glsssetabbrvfmt{\glscategory{#2}}%
2739     \glsabbrvfont{\glentryshort{#2}}#3%
2740   }%
2741 }
```

Glsxtr@p@short@

```
2742 \def\@Glsxtr@p@short@#1#2[#3]{%
2743   {%
2744     \glsssetabbrvfmt{\glscategory{#2}}%
2745     \glsabbrvfont{\Glsentryshort{#2}}#3%
2746   }%
2747 }
```

GLSxtr@p@short@

```
2748 \def\@GLSxtr@p@short@#1#2[#3]{%
2749   {%
2750     \glsssetabbrvfmt{\glscategory{#2}}%
2751     \mfirstucMakeUppercase{\glsabbrvfont{\glentryshort{#2}}#3}%
2752   }%
2753 }
```

sxtr@p@shortpl@

```
2754 \def\@glxtr@p@shortpl@#1#2[#3]{%
2755   {%
2756     \glsssetabbrvfmt{\glscategory{#2}}%
2757     \glsabbrvfont{\glentryshortpl{#2}}#3%
2758   }%
2759 }
```

Sxtr@p@shortpl@

```
2760 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
2761   {%
2762     \glsssetabbrvfmt{\glscategory{#2}}%
2763     \glsabbrvfont{\Glsentryshortpl{#2}}#3%
2764   }%
2765 }
```

GSxtr@p@shortpl@

```
2766 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
```

```

2767  {%
2768    \glsssetabbrvfmt{\glscategory{#2}}%
2769    \mfirstucMakeUppercase{\glsabbrvfont{\glstryshortpl{#2}}#3}%
2770  }%
2771 }

```

@glxtr@p@long@

```
2772 \def\@glxtr@p@long@#1#2[#3]{\glstrylong{#2}#3}}
```

@Glsxtr@p@long@

```
2773 \def\@Glsxtr@p@long@#1#2[#3]{\Glsstrylong{#2}#3}}
```

@GLSxtr@p@long@

```
2774 \def\@GLSxtr@p@long@#1#2[#3]{%
2775   {\mfirstucMakeUppercase{\glslongfont{\glstrylong{#2}}#3}}}
```

lsxtr@p@longpl@

```
2776 \def\@lsxtr@p@longpl@#1#2[#3]{\glstrylongpl{#2}#3}}
```

lSxtr@p@longpl@

```
2777 \def\@lSxtr@p@longpl@#1#2[#3]{\glslongfont{\Glsstrylongpl{#2}}#3}}
```

LSxtr@p@longpl@

```
2778 \def\@GLSxtr@p@longpl@#1#2[#3]{%
2779   {\mfirstucMakeUppercase{\glslongfont{\glstrylongpl{#2}}#3}}}
```

xtr@p@acrshort@

```
2780 \def\@gxtr@p@acrshort@#1#2[#3]{\acronymfont{\glstryshort{#2}}#3}}
```

xtr@p@acrshort@

```
2781 \def\@Glxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsstryshort{#2}}#3}}
```

xtr@p@acrshort@

```
2782 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
2783   {\mfirstucMakeUppercase{\acronymfont{\glstryshort{#2}}#3}}}
```

r@p@acrshortpl@

```
2784 \def\@gxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glstryshortpl{#2}}#3}}
```

r@p@acrshortpl@

```
2785 \def\@Glxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsstryshortpl{#2}}#3}}
```

r@p@acrshortpl@

```
2786 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
2787   {\mfirstucMakeUppercase{\acronymfont{\glstryshortpl{#2}}#3}}}
```

sxtr@p@acrlong@

```
2788 \def\@gsxtr@p@acrlong@#1#2[#3]{\glstrylong{#2}#3}}
```

sxtr@p@acrlong@

```
2789 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}
```

Sxtr@p@acrlong@

```
2790 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
2791  {\mfirstucMakeUppercase{\Glsentrylong{#2}#3}}}
```

tr@p@acrlongpl@

```
2792 \def\@glsxtr@p@acrlongpl@#1#2[#3]{\glsentrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
2793 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}
```

tr@p@acrlongpl@

```
2794 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
2795  {\mfirstucMakeUppercase{\glsentrylongpl{#2}#3}}}
```

Commands to minimise conflict.

\@glsxtrp@opt

```
2796 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}
```

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

```
2797 \newcommand*{\glsxtrsetpopts}[1]{%
2798   \renewcommand*{\@glsxtrp@opt}{#1}%
2799 }
```

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

```
2800 \newcommand*{\glossxtrsetpopts}{%
2801   \glsxtrsetpopts{noindex}%
2802 }
```

\@@glsxtrp

```
2803 \newrobustcmd*{\@@glsxtrp}[2]{%
```

Add scope.

```
2804  {%
2805    \let\glspostlinkhook\relax
2806    \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
2807  }%
2808 }
```

\@glsxtrp

```
2809 \newrobustcmd*{\@glsxtrp}[2]{%
2810   \ifcsdef{gls#1}%
2811     {%
2812       \@glsxtrp{gls#1}{#2}%
2813     }%
```

```

2814 {%
2815   \ifcsdef{glsxtr#1}%
2816   {%
2817     \@glsxtrp{glsxtr#1}{#2}%
2818   }%
2819   {%
2820     \PackageError{glossaries-extra}{‘#1’ not recognised by
2821       \string\glsxtrp}{}%
2822   }%
2823 }%
2824 }

```

\@Glsxtrp

```

2825 \newrobustcmd*{\@Glsxtrp}[2]{%
2826   \ifcsdef{Gls#1}%
2827   {%
2828     \@glsxtrp{Gls#1}{#2}%
2829   }%
2830   {%
2831     \ifcsdef{Glsxtr#1}%
2832     {%
2833       \@glsxtrp{Glsxtr#1}{#2}%
2834     }%
2835     {%
2836       \PackageError{glossaries-extra}{‘#1’ not recognised by
2837         \string\Glsxtrp}{}%
2838     }%
2839   }%
2840 }

```

\@GLSxtrp

```

2841 \newrobustcmd*{\@GLSxtrp}[2]{%
2842   \ifcsdef{GLS#1}%
2843   {%
2844     \@glsxtrp{GLS#1}{#2}%
2845   }%
2846   {%
2847     \ifcsdef{GLSxtr#1}%
2848     {%
2849       \@glsxtrp{GLSxtr#1}{#2}%
2850     }%
2851     {%
2852       \PackageError{glossaries-extra}{‘#1’ not recognised by
2853         \string\GLSxtrp}{}%
2854     }%
2855   }%
2856 }

```

\glsxtr@entry@p

```

2857 \newrobustcmd*{\glstr@headentry@p}[2]{%
2858   \gl@ifattribute{#1}{headuc}{true}%
2859   {%
2860     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
2861   }%
2862   {%
2863     \@gls@entry@field{#1}{#2}%
2864   }%
2865 }

```

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

```

2866 \ifdef\teorpdfstring
2867 {
2868   \newcommand{\glstrp}[2]{%
2869     \protect\NoCaseChange
2870     {%
2871       \protect\teorpdfstring
2872       {%
2873         \protect\glstrifinmark
2874         {%
2875           \ifcsdef{glstrhead#1}%
2876           {%
2877             {\protect\csuse{glstrhead#1}{#2}}%
2878           }%
2879           {%
2880             \glstr@headentry@p{#2}{#1}%
2881           }%
2882         }%
2883         {%
2884           \@glstrp{#1}{#2}%
2885         }%
2886       }%
2887       {%
2888         \protect\@gls@entry@field{#2}{#1}%
2889       }%
2890     }%
2891   }
2892 }
2893 {
2894   \newcommand{\glstrp}[2]{%
2895     \protect\NoCaseChange
2896     {%
2897       \protect\glstrifinmark
2898       {%
2899         \ifcsdef{glstrhead#1}%
2900         {%
2901           {\protect\csuse{glstrhead#1}}%
2902         }%
2903         {%

```

```

2904         \glxtr@headentry@p{#2}{#1}%
2905     }%
2906 }%
2907 {%
2908     \@glxtrp{#1}{#2}%
2909 }%
2910 }%
2911 }
2912 }

```

Provide short synonyms for the most common option.

`\glsp`

```

2913 \newcommand*{\glsp}{\glxtrp{short}}

```

`\glsp`

```

2914 \newcommand*{\glsp}{\glxtrp{text}}

```

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

```

2915 \ifdef\teorpdfstring
2916 {
2917     \newcommand{\Glsxtrp}[2]{%
2918         \protect\NoCaseChange
2919         {%
2920             \protect\teorpdfstring
2921             {%
2922                 \protect\glxtrifinmark
2923                 {%
2924                     \ifcsdef{Glsxtrhead#1}%
2925                     {%
2926                         {\protect\csuse{Glsxtrhead#1}{#2}}%
2927                     }%
2928                     {%
2929                         \protect\@Gls@entry@field{#2}{#1}%
2930                     }%
2931                 }%
2932             }%
2933             \@Glsxtrp{#1}{#2}%
2934         }%
2935     }%
2936     {%
2937         \protect\@gl@entry@field{#2}{#1}%
2938     }%
2939 }%
2940 }
2941 }
2942 {
2943     \newcommand{\Glsxtrp}[2]{%

```

```

2944 \protect\NoCaseChange
2945 {%
2946 \protect\glxtrifinmark
2947 {%
2948 \ifcsdef{Glsxtrhead#1}%
2949 {%
2950 {\protect\csuse{Glsxtrhead#1}}%
2951 }%
2952 {%
2953 \protect\@Gls@entry@field{#2}{#1}%
2954 }%
2955 }%
2956 {%
2957 \@Glsxtrp{#1}{#2}%
2958 }%
2959 }%
2960 }
2961 }

```

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

```

2962 \ifdef\texorpdfstring
2963 {
2964 \newcommand{\GLSxtrp}[2]{%
2965 \protect\NoCaseChange
2966 {%
2967 \protect\texorpdfstring
2968 {%
2969 \protect\glxtrifinmark
2970 {%
2971 \ifcsdef{GLSxtr#1}%
2972 {%
2973 {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
2974 }%
2975 {%
2976 \protect\mfirstucMakeUppercase
2977 {%
2978 \protect\@gls@entry@field{#2}{#1}%
2979 }%
2980 }%
2981 }%
2982 {%
2983 \@GLSxtrp{#1}{#2}%
2984 }%
2985 }%
2986 {%
2987 \protect\@gls@entry@field{#2}{#1}%
2988 }%
2989 }%
2990 }

```

```

2991 }
2992 {
2993   \newcommand{\GLSxtrp}[2]{%
2994     \protect\NoCaseChange
2995     {%
2996       \protect\glxtrifinmark
2997       {%
2998         \ifcsdef{GLSxtr#1}%
2999         {%
3000           {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3001         }%
3002       {%
3003         \protect\mfirstucMakeUppercase
3004         {%
3005           \protect\@gls@entry@field{#2}{#1}%
3006         }%
3007       }%
3008     }%
3009   {%
3010     \@GLSxtrp{#1}{#2}%
3011   }%
3012 }%
3013 }
3014 }

```

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.

```

3015 \renewcommand*\@glsunset[1]{%
3016   \@glsunset{#1}%
3017   \glxtrpostunset{#1}%
3018 }%

```

glxtrpostunset

```

3019 \newcommand*\glxtrpostunset[1]{%

```

\@glslocalunset Local unset.

```

3020 \renewcommand*\@glslocalunset[1]{%
3021   \@glslocalunset{#1}%
3022   \glxtrpostlocalunset{#1}%

```

```

3023 }%

rpostlocalunset
3024 \newcommand*{\glxtrpostlocalunset}[1]{}

\@glsreset   Global reset.
3025 \renewcommand*{\@glsreset}[1]{%
3026   \@@glsreset{#1}%
3027   \glxtrpostreset{#1}%
3028 }%

glxtrpostreset
3029 \newcommand*{\glxtrpostreset}[1]{}

\@glslocalreset   Local reset.
3030 \renewcommand*{\@glslocalreset}[1]{%
3031   \@@glslocalreset{#1}%
3032   \glxtrpostlocalreset{#1}%
3033 }%

rpostlocalreset
3034 \newcommand*{\glxtrpostlocalreset}[1]{}

leEntryCounting   The first argument is the list of categories and the second argument is the value of the en-
                    trycount attribute.
3035 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
    Enable entry counting:
3036   \glsenableentrycount
    Redefine \gls etc:
3037   \renewcommand*{\gls}{\cglss}%
3038   \renewcommand*{\Gls}{\cGls}%
3039   \renewcommand*{\glspl}{\cglsspl}%
3040   \renewcommand*{\Glspl}{\cGlspl}%
3041   \renewcommand*{\GLS}{\cGLS}%
3042   \renewcommand*{\GLSpl}{\cGLSpl}%
    Set the entrycount attribute:
3043   \@glxtr@setentrycountunsetattr{#1}{#2}%
    In case this command is used again:
3044   \let\GlsXtrEnableEntryCounting\@glxtr@setentrycountunsetattr
3045   \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
3046     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3047       can't be used with \string\GlsXtrEnableEntryCounting}%
3048     {Use one or other but not both commands}}%
3049 }

```

countunsetattr

```
3050 \newcommand*{\@glxtr@setentrycountunsetattr}[2]{%
3051   \@for\@glxtr@cat:=#1\do
3052   {%
3053     \ifdefempty{\@glxtr@cat}{}%
3054     {%
3055       \glsssetcategoryattribute{\@glxtr@cat}{entrycount}{#2}%
3056     }%
3057   }%
3058 }
```

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

enableentrycount

```
3059 \renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
3060   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
```

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

```
3061   \renewcommand*{\gl@defdocnewglossaryentry}{%
3062     \renewcommand*{\newglossaryentry}[2]{%
3063       \PackageError{glossaries}{\string\newglossaryentry\space
3064         may only be used in the preamble when entry counting has
3065         been activated}{If you use \string\glsenableentrycount\space
3066         you must place all entry definitions in the preamble not in
3067         the document environment}%
3068     }%
3069   }%
```

New commands to access new fields:

```
3070   \newcommand*{\glsetentrycurrcount}[1]{%
3071     \ifcsundef{glo@\glsetoklabel{##1}@currcount}%
3072     {0}{\@gl@entry@field{##1}{currcount}}%
3073   }%
3074   \newcommand*{\glsetentryprevcount}[1]{%
3075     \ifcsundef{glo@\glsetoklabel{##1}@prevcount}%
3076     {0}{\@gl@entry@field{##1}{prevcount}}%
3077   }%
```

Adjust post unset and reset:

```
3078   \let\@glxtr@entrycount@org@unset\glxtrpostunset
3079   \renewcommand*{\glxtrpostunset}[1]{%
3080     \@glxtr@entrycount@org@unset{##1}%
3081     \@gl@increment@currcount{##1}%
3082   }%
3083   \let\@glxtr@entrycount@org@localunset\glxtrpostlocalunset
3084   \renewcommand*{\glxtrpostlocalunset}[1]{%
3085     \@glxtr@entrycount@org@localunset{##1}%
3086     \@gl@local@increment@currcount{##1}%
3087   }%
```

```

3088 \let\@glxstr@entrycount@org@reset\glxstrpostreset
3089 \renewcommand*{\glxstrpostreset}[1]{%
3090   \@glxstr@entrycount@org@reset{##1}%
3091   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
3092 }%
3093 \let\@glxstr@entrycount@org@localreset\glxstrpostlocalreset
3094 \renewcommand*{\glxstrpostlocalreset}[1]{%
3095   \@glxstr@entrycount@org@localreset{##1}%
3096   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
3097 }%

```

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

```

3098 \let\@cgls@\@cgls@
3099 \let\@cglspl@\@cglspl@

3100 \let\@cGls@\@cGls@
3101 \let\@cGlspl@\@cGlspl@
3102 \let\@cGLS@\@cGLS@
3103 \let\@cGLSpl@\@cGLSpl@

```

The rest is as the original definition.

```

3104 \AtEndDocument{\@gls@write@entrycounts}%
3105 \renewcommand*{\@gls@entry@count}[2]{%
3106   \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
3107 }%
3108 \let\glsenableentrycount\relax
3109 \renewcommand*{\glsenableentryunitcount}{%
3110   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
3111     can't be used with \string\glsenableentrycount}%
3112   {Use one or other but not both commands}%
3113 }%
3114 }

```

`@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.

```

3115 \renewcommand*{\@gls@write@entrycounts}{%
3116   \immediate\write\@auxout
3117     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
3118   \count@=0\relax
3119   \forallglsentries{\@glsentry}{%
3120     \gls@hasattribute{\@glsentry}{entrycount}%
3121     {%
3122       \ifglsused{\@glsentry}%
3123       {%
3124         \immediate\write\@auxout
3125           {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
3126       }%
3127     }%
3128   \advance\count@ by \@ne

```

```

3129 }%
3130 {}%
3131 }%
3132 \ifnum\count@=0
3133 \GlossariesExtraWarningNoLine{Entry counting has been enabled
3134 \MessageBreak with \string\glsenableentrycount\space but the
3135 \MessageBreak attribute ‘entrycount’ hasn’t
3136 \MessageBreak been assigned to any of the defined
3137 \MessageBreak entries}%
3138 \fi
3139 }

```

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

```

3140 \newcommand*{\glxtrifcounttrigger}[3]{%
3141 \glshasattribute{#1}{entrycount}%
3142 {%
3143 \ifnum\gl Sentryprevcount{#1}>\gl sgetattribute{#1}{entrycount}\relax
3144 #3%
3145 \else
3146 #2%
3147 \fi
3148 }%
3149 {#3}%
3150 }

```

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

`\@@cgl`s@

```

3151 \def\@@cgl s@#1#2[#3]{%
3152 \glxtrifcounttrigger{#2}%
3153 {%
3154 \cgl sformat{#2}{#3}%
3155 \gl sunset{#2}%
3156 }%
3157 {%
3158 \@cgl s@{#1}{#2}[#3]%
3159 }%
3160 }%

```

`\@@cgl spl`@

```

3161 \def\@@cgl spl@#1#2[#3]{%
3162 \glxtrifcounttrigger{#2}%
3163 {%
3164 \cgl splformat{#2}{#3}%
3165 \gl sunset{#2}%

```

```

3166 }%
3167 {%
3168     \@glsp1@{#1}-{#2}[#3]%
3169 }%
3170 }%

```

\@@cGls@

```

3171 \def\@@cGls@#1#2[#3]{%
3172     \glxtrifcounttrigger{#2}%
3173     {%
3174         \cGlsformat{#2}{#3}%
3175         \glset{#2}%
3176     }%
3177     {%
3178         \@Gls@{#1}-{#2}[#3]%
3179     }%
3180 }%

```

\@@cGlsp1@

```

3181 \def\@@cGlsp1@#1#2[#3]{%
3182     \glxtrifcounttrigger{#2}%
3183     {%
3184         \cGlsp1format{#2}{#3}%
3185         \glset{#2}%
3186     }%
3187     {%
3188         \@Glsp1@{#1}-{#2}[#3]%
3189     }%
3190 }%

```

\@@cGLS@

```

3191 \def\@@cGLS@#1#2[#3]{%
3192     \glxtrifcounttrigger{#2}%
3193     {%
3194         \cGLSformat{#2}{#3}%
3195         \glset{#2}%
3196     }%
3197     {%
3198         \@GLS@{#1}-{#2}[#3]%
3199     }%
3200 }%

```

\@@cGLSp1@

```

3201 \def\@@cGLSp1@#1#2[#3]{%
3202     \glxtrifcounttrigger{#2}%
3203     {%
3204         \cGLSp1format{#2}{#3}%
3205         \glset{#2}%
3206     }%

```

```

3207  {%
3208    \@GLSp1@{#1}{#2}[#3]%
3209  }%
3210 }%

```

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

```

\@cgl's@
3211 \def\@cgl's@#1#2[#3]{\@gls@{#1}{#2}[#3]}

```

```

\@cGls@
3212 \def\@cGls@#1#2[#3]{\@Gls@{#1}{#2}[#3]}

```

```

\@cgl'spl@
3213 \def\@cgl'spl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}

```

```

\@cGlspl@
3214 \def\@cGlspl@#1#2[#3]{\@Glspl@{#1}{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

```

\cGLS
3215 \newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}

```

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

```

3216 \newcommand*{\@cGLS}[2][{}]{%
3217   \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[{}]}%
3218 }

```

```

\@cGLS@
3219 \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.

```

3220 \newcommand*{\cGLSformat}[2]{%
3221   \expandafter\mfirstucMakeUppercase\expandafter{\cgl'sformat{#1}{#2}}%
3222 }

```

```

\cGLSp1
3223 \newrobustcmd*{\cGLSp1}{\@gls@hyp@opt\@cGLSp1}

```

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

```

3224 \newcommand*{\@cGLSp1}[2][{}]{%
3225   \new@ifnextchar[{\@cGLSp1@{#1}{#2}}{\@cGLSp1@{#1}{#2}[{}]}%
3226 }

```

```
\@cGLSp1@
3227 \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.

```
3228 \newcommand*\cGLSplformat}[2]{%
3229 \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}}%
3230 }
```

Modify the trigger formats to check for the regular attribute.

\cglformat

```
3231 \renewcommand*\cglformat}[2]{%
3232 \glsifregular{#1}
3233 {\glsentryfirst{#1}}%
3234 {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
3235 }
```

\cGlsformat

```
3236 \renewcommand*\cGlsformat}[2]{%
3237 \glsifregular{#1}
3238 {\Glsentryfirst{#1}}%
3239 {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
3240 }
```

\cglsplformat

```
3241 \renewcommand*\cglsplformat}[2]{%
3242 \glsifregular{#1}
3243 {\glsentryfirstplural{#1}}%
3244 {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
3245 }
```

\cGlsplformat

```
3246 \renewcommand*\cGlsplformat}[2]{%
3247 \glsifregular{#1}
3248 {\Glsentryfirstplural{#1}}%
3249 {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}%#2%
3250 }
```

New code similar to above for unit counting.

defunitcounters

```
3251 \newcommand*\@@newglossaryentry@defunitcounters{%
3252 \edef\@glo@countunit{\csuse{@glsxtr@categoryattr@@\@glo@category @unitcount}}%
3253 \ifdefvoid\@glo@countunit
3254 {}%
3255 {%
3256 \@glsxtr@ifunitcounter{\@glo@countunit}%
```

```

3257     {}%
3258     {\expandafter\@glsxtr@addunitcounter\expandafter{\@glo@countunit}}}%
3259 }%
3260 }

```

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

```

3261 \newcommand*{\@glsxtr@unitcountlist}{}

```

`@addunitcounter`

```

3262 \newcommand*{\@glsxtr@addunitcounter}[1]{%
3263   \listadd{\@glsxtr@unitcountlist}{#1}%
3264   \ifcsundef{glsxtr@theunit@#1}
3265   {%
3266     \ifcsdef{theH#1}%
3267     {\csdef{glsxtr@theunit@#1}{\csuse{theH#1}}}%
3268     {\csdef{glsxtr@theunit@#1}{\csuse{the#1}}}%
3269   }%
3270   {}%
3271 }

```

`r@ifunitcounter`

```

3272 \newcommand*{\@glsxtr@ifunitcounter}[3]{%
3273   \xifinlist{#1}{\@glsxtr@unitcountlist}{#2}{#3}%
3274 }

```

`urrentunitcount`

```

3275 \newcommand*\@glsxtr@currentunitcount[1]{%
3276   glo@\glsdetoklabel{#1}@currunit@\glsgetattribute{#1}{unitcount}.%
3277   \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3278 }

```

`eviousunitcount`

```

3279 \newcommand*\@glsxtr@previousunitcount[1]{%
3280   glo@\glsdetoklabel{#1}@prevunit@\glsgetattribute{#1}{unitcount}.%
3281   \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3282 }

```

`t@currunitcount`

```

3283 \newcommand*{\@gls@increment@currunitcount}[1]{%
3284   \glsattribute{#1}{unitcount}%
3285   {%
3286     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3287     \ifcsundef{\@glsxtr@csname}%
3288     {%
3289       \csgdef{\@glsxtr@csname}{1}%
3290       \listcsxadd
3291       {glo@\glsdetoklabel{#1}@unitlist}%
3292       {\glsgetattribute{#1}{unitcount}.%

```

```

3293         \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
3294     }%
3295 }%
3296 {%
3297     \csxdef{\@glxstr@csname}%
3298     {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
3299 }%
3300 }%
3301 {}%
3302 }

```

t@currunitcount

```

3303 \newcommand*{\@glsl@local@increment@currunitcount}[1]{%
3304     \glshasattribute{#1}{unitcount}%
3305     {%
3306         \edef\@glxstr@csname{\@glxstr@currentunitcount{#1}}%
3307         \ifcsundef{\@glxstr@csname}%
3308         {%
3309             \csdef{\@glxstr@csname}{1}%
3310             \listcseadd
3311             {glo@\glsdetoklabel{#1}@unitlist}%
3312             {\glsggetattribute{#1}{unitcount}.%
3313             \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
3314         }%
3315     }%
3316     {%
3317         \csedef{\@glxstr@csname}%
3318         {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
3319     }%
3320 }%
3321 {}%
3322 }

```

r@currunitcount

```

3323 \newcommand*{\@glxstr@currunitcount}[2]{%
3324     \ifcsundef
3325     {glo@\glsdetoklabel{#1}@currunit@#2}%
3326     {0}%
3327     {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
3328 }%

```

r@prevunitcount

```

3329 \newcommand*{\@glxstr@prevunitcount}[2]{%
3330     \ifcsundef
3331     {glo@\glsdetoklabel{#1}@prevunit@#2}%
3332     {0}%
3333     {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}%
3334 }%

```

entryunitcount

```
3335 \newcommand*{\glsenableentryunitcount}{%
```

Enable new fields:

```
3336 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
```

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

```
3337 \renewcommand*{\gls@defdocnewglossaryentry}{%
```

```
3338 \renewcommand*{\newglossaryentry[2]{%
```

```
3339 \PackageError{glossaries}{\string\newglossaryentry\space
```

```
3340 may only be used in the preamble when entry counting has
```

```
3341 been activated}{If you use \string\glsenableentryunitcount\space
```

```
3342 you must place all entry definitions in the preamble not in
```

```
3343 the document environment}%
```

```
3344 }%
```

```
3345 }%
```

New commands to access new fields:

```
3346 \newcommand*{\glsentrycurrcount}[1]{%
```

```
3347 \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
```

```
3348 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
```

```
3349 }%
```

```
3350 \newcommand*{\glsentryprevcount}[1]{%
```

```
3351 \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
```

```
3352 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
```

```
3353 }%
```

Access total count:

```
3354 \newcommand*{\glsentryprevtotalcount}[1]{%
```

```
3355 \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
```

```
3356 {0}%
```

```
3357 {%
```

```
3358 \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
```

```
3359 }%
```

```
3360 }%
```

Access max value:

```
3361 \newcommand*{\glsentryprevmaxcount}[1]{%
```

```
3362 \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
```

```
3363 {0}%
```

```
3364 {%
```

```
3365 \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
```

```
3366 }%
```

```
3367 }%
```

Adjust post unset and reset:

```
3368 \let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
```

```
3369 \renewcommand*{\glsxtrpostunset}[1]{%
```

```
3370 \@glsxtr@entryunitcount@org@unset{##1}%
```

```
3371 \@gls@increment@currunitcount{##1}%
```

```
3372 }%
```

```
3373 \let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
```

```

3374 \renewcommand*{\glxtrpostlocalunset}[1]{%
3375   \@glxtr@entryunitcount@org@localunset{##1}%
3376   \@glx@local@increment@currunitcount{##1}%
3377 }%
3378 \let\@glxtr@entryunitcount@org@reset\glxtrpostreset
3379 \renewcommand*{\glxtrpostreset}[1]{%
3380   \glshasattribute{##1}{unitcount}%
3381   {%
3382     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
3383     \ifcsundef{\@glxtr@csname}%
3384     {}%
3385     {\csgdef{\@glxtr@csname}{0}}%
3386   }%
3387   {}%
3388 }%
3389 \let\@glxtr@entryunitcount@org@localreset\glxtrpostlocalreset
3390 \renewcommand*{\glxtrpostlocalreset}[1]{%
3391   \@glxtr@entryunitcount@org@localreset{##1}%
3392   \glshasattribute{##1}{unitcount}%
3393   {%
3394     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
3395     \ifcsundef{\@glxtr@csname}%
3396     {}%
3397     {\csdef{\@glxtr@csname}{0}}%
3398   }%
3399   {}%
3400 }%

```

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

```

3401 \let\@cgl@\@cgl@
3402 \let\@cgl@pl@\@cgl@pl@

3403 \let\@cGls@\@cGls@
3404 \let\@cGlspl@\@cGlspl@
3405 \let\@cGLS@\@cGLS@
3406 \let\@cGLSpl@\@cGLSpl@

```

Write information to the aux file.

```

3407 \AtEndDocument{\@gl@write@entryunitcounts}%
3408 \renewcommand*{\@gl@entry@unitcount}[3]{%
3409   \csgdef{glo@gl@sdetoklabel{##1}@prevunit@##3}{##2}%
3410   \ifcsundef{glo@gl@sdetoklabel{##1}@prevunittotal}%
3411   {\csgdef{glo@gl@sdetoklabel{##1}@prevunittotal}{##2}}%
3412   {%
3413     \csxdef{glo@gl@sdetoklabel{##1}@prevunittotal}{
3414       \number\numexpr\csuse{glo@gl@sdetoklabel{##1}@prevunittotal}+##2}%
3415     }%
3416     \ifcsundef{glo@gl@sdetoklabel{##1}@prevunitmax}%
3417     {\csgdef{glo@gl@sdetoklabel{##1}@prevunitmax}{##2}}%

```

```

3418   {%
3419     \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
3420       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
3421     \fi
3422   }%
3423 }%
3424 \let\glsenableentryunitcount\relax
3425 \renewcommand*{\glsenableentrycount}{%
3426   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
3427     can't be used with \string\glsenableentryunitcount}%
3428   {Use one or other but not both commands}%
3429 }%
3430 }
3431 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

3432 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

3433 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
3434   \immediate\write\@auxout
3435   {\string\@gls@entry@unitcount
3436     {\@glsentry}%
3437     {\@glsxtr@currunitcount{\@glsentry}{#1}%
3438     }%
3439     {#1}}%
3440 }

```

entryunitcounts

```

3441 \newcommand*{\@gls@write@entryunitcounts}{%
3442   \immediate\write\@auxout
3443   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
3444   \count@=0\relax
3445   \forallglsentries{\@glsentry}{%
3446     \gls@hasattribute{\@glsentry}{unitcount}%
3447     {%
3448       \ifglsused{\@glsentry}%
3449       {%
3450         \forlistcsloop
3451           {\@gls@write@entryunitcounts@do}%
3452           {glo@\glsdetoklabel{\@glsentry}@unitlist}%
3453       }%
3454     }%
3455     \advance\count@ by \@ne
3456   }%
3457   {}%
3458 }%
3459 \ifnum\count@=0
3460   \GlossariesExtraWarningNoLine{Entry counting has been enabled

```

```

3461 \MessageBreak with \string\glsenableentryunitcount\space but the
3462 \MessageBreak attribute ‘unitcount’ hasn’t
3463 \MessageBreak been assigned to any of the defined
3464 \MessageBreak entries}%
3465 \fi
3466 }

```

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.

```

3467 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

3468 \glsenableentryunitcount

```

Redefine \gls etc:

```

3469 \renewcommand*{\gls}{\cglsl}%
3470 \renewcommand*{\Gls}{\cGls}%
3471 \renewcommand*{\glspl}{\cglspl}%
3472 \renewcommand*{\Glspl}{\cGlspl}%
3473 \renewcommand*{\GLS}{\cGLS}%
3474 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3475 \@glstxr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

3476 \let\GlsXtrEnableEntryUnitCounting\@glstxr@setentryunitcountunsetattr
3477 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3478 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3479 can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
3480 {Use one or other but not both commands}}%
3481 }

```

countunsetattr

```

3482 \newcommand*{\@glstxr@setentryunitcountunsetattr}[3]{%
3483 \@for\@glstxr@cat:=#1\do
3484 {%
3485 \ifdefempty{\@glstxr@cat}{}%
3486 {%
3487 \glssetcategoryattribute{\@glstxr@cat}{entrycount}{#2}%
3488 \glssetcategoryattribute{\@glstxr@cat}{unitcount}{#3}%
3489 }%
3490 }%
3491 }

```

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`

```

3492 \renewcommand*{\SetGenericNewAcronym}{%
3493   \let\@Gls@entryname\@Gls@acentryname
3494   \renewcommand{\newacronym}[4][\%
3495     \ifdefempty{\@glsacronymlists}%
3496     {%
3497       \def\@glo@type{\acronymtype}%
3498       \setkeys{glossentry}{##1}%
3499       \DeclareAcronymList{\@glo@type}%
3500     }%
3501     {}%
3502     \glskeylisttok{##1}%
3503     \glslabeltok{##2}%
3504     \glsshorttok{##3}%
3505     \glslongtok{##4}%
3506     \newacronymhook
3507     \protected@edef\@do@newglossaryentry{%
3508       \noexpand\newglossaryentry{\the\glslabeltok}%
3509       {%
3510         type=\acronymtype,%
3511         name={\expandonce{\acronymentry{##2}}},%
3512         sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
3513         text={\the\glsshorttok},%
3514         short={\the\glsshorttok},%
3515         shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3516         long={\the\glslongtok},%
3517         longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3518         category=acronym,%
3519         \GenericAcronymFields,%
3520         \the\glskeylisttok
3521       }%
3522     }%
3523     \@do@newglossaryentry
3524   }%
3525   \renewcommand*{\acrfullfmt}[3]{%
3526     \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
3527   \renewcommand*{\Acrfullfmt}[3]{%
3528     \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
3529   \renewcommand*{\ACRfullfmt}[3]{%
3530     \glslink[##1]{##2}{%
3531       \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
3532   \renewcommand*{\acrfullplfmt}[3]{%
3533     \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
3534   \renewcommand*{\Acrfullplfmt}[3]{%
3535     \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%

```

```

3536 \renewcommand*{\ACRfullplfmt}[3]{%
3537   \glslink{##1}{##2}}{%
3538   \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
3539 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
3540 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
3541 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
3542 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
3543 }

```

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:

```

3544 \let\@glxtr@org@setacronymstyle\setacronymstyle
3545 \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`.

```

3546 \newcommand*{\MakeAcronymsAbbreviations}{%
3547   \renewcommand*{\newacronym}[4][1]{%
3548     \glxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
3549   }%
3550   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
3551   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
3552   \renewcommand*{\setacronymstyle}[1]{%
3553     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}}
3554     unavailable.
3555     Use \string\setabbreviationstyle\space instead.
3556     The original acronym interface can be restored with
3557     \string\RestoreAcronyms}{}%
3558   }%
3559   \renewcommand*{\newacronymstyle}[1]{%
3560     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
3561     available unless you restore the original acronym interface with
3562     \string\RestoreAcronyms}%
3563     \@glxtr@org@newacronymstyle{##1}%
3564   }%
3565 }

```

Switch acronyms to abbreviations:

```

3566 \MakeAcronymsAbbreviations

```

RestoreAcronyms Restore acronyms to glossaries interface.

```

3567 \newcommand*{\RestoreAcronyms}{%
3568   \SetGenericNewAcronym
3569   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
3570   \renewcommand*{\acronymfont}[1]{##1}%
3571   \let\setacronymstyle\@glxtr@org@setacronymstyle
3572   \let\newacronymstyle\@glxtr@org@newacronymstyle

```

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

```

3573 \renewcommand*\@gls@link@checkfirsthyper{%
3574   \ifglused{\glslabel}%
3575   {\let\glstrifwasfirstuse\@secondoftwo}
3576   {\let\glstrifwasfirstuse\@firstoftwo}%
3577   \@glstr@org@checkfirsthyper
3578 }
3579 \glssetcategoryattribute{acronym}{regular}{false}%
3580 \setacronymstyle{long-short}%
3581 }

```

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

```

3582 \renewcommand*\glsacspace[1]{%
3583   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
3584   \ifdim\dimen@<\glsacspacemax~\else\space\fi
3585 }

```

`\glsacspacemax` Value used in the above.

```

3586 \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`

```

3587 \newcommand*\@glstr@reg@glosslist{}

```

Save the original definition of `\makeglossaries`:

```

3588 \let\@glstr@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`

```

3589 \renewcommand*\makeglossaries[1][]{%
3590   \ifx\@glstr@record@setting\@glstr@record@setting@only
3591     \PackageError{glossaries-extra}{\string\makeglossaries\space
3592       not permitted\MessageBreak with record=only package option}%
3593     {You may only use \string\makeglossaries\space with
3594       record=off or record=alsoindex options}%
3595   \else
3596     \ifblank{#1}%
3597     {\@glstr@org@makeglossaries}%

```

```

3598 {%
3599 \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
3600 \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
3601 not permitted\MessageBreak with record=alsoindex package option}%
3602 {You may only use the hybrid \string\makeglossaries[...]\space with
3603 record=off option}%
3604 \else
3605 \edef\@glxtr@reg@glosslist{#1}%
3606 \ifundef{\glswrite}{\newwrite\glswrite}{}%
3607 \protected@write\@auxout{}{\string\providecommand
3608 \string\@glxtr@reg@glosslist{#1}}
3609 \protected@write\@auxout{}{\string\providecommand
3610 \string\@glxtr@reg@glosslist{#1}}
3611 \protected@write\@auxout{}{\string\@glxtr@reg@glosslist{#1}}
3612 \protected@write\@auxout{}{\string\@glxtr@reg@glosslist{#1}}
3613 \protected@write\@auxout{}{\string\@glxtr@reg@glosslist{#1}}
3614 \write\@auxout{\string\providecommand\string\@glxtr@reg@glosslist{#1}}

```

Iterate through each supplied glossary type and activate it.

```

3615 \for\@glo@type:=#1\do{%
3616 \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
3617 }%

```

New glossaries must be created before \makeglossaries:

```

3618 \renewcommand*\newglossary[4][]{%
3619 \PackageError{glossaries}{New glossaries
3620 must be created before \string\makeglossaries}{You need
3621 to move \string\makeglossaries\space after all your
3622 \string\newglossary\space commands}}%

```

Any subsequence instances of this command should have no effect

```

3623 \let\@makeglossary\relax
3624 \let\makeglossary\relax
3625 \renewcommand\makeglossaries[1][]{%

```

Disable all commands that have no effect after \makeglossaries

```

3626 \@disable@onlypremakeg

```

Allow see key:

```

3627 \let\gls@checkseeallowed\relax

```

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

```

3628 \renewcommand*\@do@seeglossary[2]{%
3629 \glsdoifexists{##1}%
3630 {%
3631 \edef\@gls@label{\glsdetoklabel{##1}}%
3632 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3633 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glxtr@reg@glosslist}%
3634 {\@glxtr@org@doseeglossary{##1}{##2}}%
3635 }%
3636 \@glxtrwrglossmark

```

```

3637         \protected@write\@auxout{}\{-%
3638         \string\@gls@reference
3639         {\@gls@type}\{\@gls@label}\{\string\glsseeformat##2\}}}%
3640     }%
3641 }%
3642 }%
3643 }%

Adjust \@do@wrglossary
3644 \let\@glsxtr@do@wrglossary\@do@wrglossary
3645 \def\@do@wrglossary{%
3646     \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3647     \expandafter\DTLifinlist\expandafter{\@gls@type}\{\@glsxtr@reg@glosslist}%
3648     {\@glsxtr@do@wrglossary}%
3649     {\@gls@noidxglossary}%
3650 }%

Suppress warning about no \makeglossaries
3651 \let\warn@nomakeglossaries\relax
3652 \def\warn@noprintglossary{%
3653     \GlossariesWarningNoLine{No \string\printglossary\space
3654     or \string\printglossaries\space
3655     found.^^J(Remove \string\makeglossaries\space if you don't want
3656     any glossaries.)^^JThis document will not have a glossary}%
3657 }%

Only warn for glossaries not listed.
3658 \renewcommand{\@gls@noref@warn}[1]{%
3659     \edef\@gls@type{##1}%
3660     \expandafter\DTLifinlist\expandafter{\@gls@type}\{\@glsxtr@reg@glosslist}%
3661     {%
3662         \GlossariesExtraWarning{Can't use
3663         \string\printnoidxglossary[type={\@gls@type}]
3664         when '\@gls@type' is listed in the optional argument of
3665         \string\makeglossaries}%
3666     }%
3667     {%
3668         \GlossariesWarning{Empty glossary for
3669         \string\printnoidxglossary[type={##1}].
3670         Rerun may be required (or you may have forgotten to use
3671         commands like \string\gls)}}%
3672     }%
3673 }%

Adjust display number list to check for type:
3674 \renewcommand*\@glsdisplaynumberlist[1]{%
3675     \expandafter\DTLifinlist\expandafter{##1}\{\@glsxtr@reg@glosslist}%
3676     {\@glsxtr@idx@displaynumberlist{##1}}%
3677     {\@glsxtr@noidx@displaynumberlist{##1}}%
3678 }%

```

Adjust entry list:

```

3679      \renewcommand*{\glsentrynumberlist}[1]{%
3680          \expandafter\DTLifinlist\expandafter{##1}{\@glstr@reg@glosslist}%
3681          {\@glstr@idx@entrynumberlist{##1}}%
3682          {\@glstr@noidx@entrynumberlist{##1}}%
3683      }%

```

Adjust number list loop

```

3684      \renewcommand*{\glsnumberlistloop}[2]{%
3685          \expandafter\DTLifinlist\expandafter{##1}{\@glstr@reg@glosslist}%
3686          {%
3687              \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3688                  not available for glossary ‘##1’}{}%
3689          }%
3690          {\@glstr@noidx@numberlistloop{##1}{##2}}%
3691      }%

```

Only sanitize sort for normal indexing glossaries.

```

3692      \renewcommand*{\glsprestandardsort}[3]{%
3693          \expandafter\DTLifinlist\expandafter{##2}{\@glstr@reg@glosslist}%
3694          {%
3695              \glsdosanitizesort
3696          }%
3697          {%
3698              \ifglssanitizesort
3699                  \@glstr@noidx@sanitizesort
3700              \else
3701                  \@glstr@noidx@nosanitizesort
3702              \fi
3703          }%
3704      }%

```

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

```

3705      \renewcommand*\new@glossaryentry[2]{%
3706          \PackageError{glossaries-extra}{Glossary entries must be defined
3707              in the preamble\MessageBreak when you use the optional argument
3708              of \string\makeglossaries}{Either move your definitions to the
3709              preamble or don't use the optional argument of
3710              \string\makeglossaries}%
3711      }%

```

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).

```

3712      \let\@glo@assign@sortkey\@glstr@mixed@assign@sortkey
3713      \renewcommand*{\@printgloss@setsort}{%

```

Need to extract just the type value.

```

3714          \expandafter\@glstr@gettype\expandafter,\@glstr@printglossopts,%
3715          type=\glsdefaulttype,\@end@glstr@gettype
3716      \def\@glo@sorttype{\@glo@default@sorttype}%

```

3717 }%

Check automake setting:

```
3718 \ifglautomake
3719 \renewcommand*{\@gls@doautomake}{%
3720 \@for\@gls@type:=\@glxtr@reg@glosslist\do{%
3721 \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
3722 }%
3723 }%
3724 \fi
```

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

```
3725 \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
3726 \fi
3727 }%
3728 \fi
3729 }
```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

`rgprintglossary` This no longer simply saves `\@printglossary` with `\let` but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes `\provideignoredglossary` to the `glstex` file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```
3730 \newcommand{\@glxtr@orgprintglossary}[2]{%
3731 \def\@glo@type{\glsdefaulttype}%
```

Add check here.

```
3732 \def\glossarytitle{%
3733 \ifcsdef{\@glo@type\@glo@type @title}%
3734 {\csuse{\@glo@type\@glo@type @title}}%
3735 {\glossaryname}}%
3736 \def\glossarytoctitle{\glossarytitle}%
3737 \let\org@glossarytitle\glossarytitle
3738 \def\@glossarystyle{%
3739 \ifx\@glossary@default@style\relax
3740 \GlossariesWarning{No default glossary style provided \MessageBreak
3741 for the glossary '\@glo@type'. \MessageBreak
3742 Using deprecated fallback. \MessageBreak
3743 To fix this set the style with \MessageBreak
3744 \string\setglossarystyle\space or use the \MessageBreak
3745 style key=value option}%
3746 \fi
3747 }%
3748 \def\gls@dotoc@title{\gls@set@toc@title{\@glo@type}}%
3749 \let\@org@glossaryentrynumbers\glossaryentrynumbers
3750 \bgroup
3751 \@printgloss@setsort
3752 \setkeys{printgloss}{#1}%
```

```

3753 \ifx\glossarytitle\org@glossarytitle
3754 \else
3755 \cslet{@glotype@}@glo@type @title-{\glossarytitle}%
3756 \fi
3757 \let\currentglossary@glo@type
3758 \let\org@glossaryentrynumbers\glossaryentrynumbers
3759 \let\glsnonextpages@glsnonextpages
3760 \let\glsnextpages@glsnextpages

3761 \glsxtractivatenopost
3762 \gls@dotoc@title
3763 \@glossarystyle
3764 \let\gls@org@glossaryentryfield\glossentry
3765 \let\gls@org@glossarysubentryfield\subglossentry
3766 \renewcommand{\glossentry}[1]{%
3767 \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
3768 \gls@org@glossaryentryfield{##1}%
3769 }%
3770 \renewcommand{\subglossentry}[2]{%
3771 \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
3772 \gls@org@glossarysubentryfield{##1}{##2}%
3773 }%
3774 \@gls@preglossaryhook
3775 #2%
3776 \egroup
3777 \global\let\glossaryentrynumbers\org@glossaryentrynumbers
3778 \global\let\warn@noprintglossary\relax
3779 }

```

ractivatenopost Change \nopostdesc and \glsxtrnopostpunc to behave as they do in the glossary.

```

3780 \newcommand*{\glsxtractivatenopost}{%
3781 \let\nopostdesc\@nopostdesc
3782 \let\glsxtrnopostpunc@glsxtr@nopostpunc
3783 }

```

lsxtrnopostpunc

```

3784 \newrobustcmd*{\glsxtrnopostpunc}{%

```

sxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```

3785 \newcommand{\@glsxtr@nopostpunc}{%
3786 \let\@@glsxtr@org@postdescription\glspostdescription
3787 \ifglsnopostdot
3788 \renewcommand{\glspostdescription}{%
3789 \glsnopostdottrue
3790 \let\glspostdescription\@@glsxtr@org@postdescription
3791 \let\glsxtrrestorepostpunc@glsxtr@restore@postpunc
3792 \glsxtrpostdescription
3793 \@glsxtr@nopostpunc@postdesc}%

```

```

3794 \else
3795   \renewcommand{\glspostdescription}{%
3796     \let\glspostdescription\@glxtr@org@postdescription
3797     \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
3798     \glxtrpostdescription
3799     \@glxtr@nopostpunc@postdesc}%
3800 \fi
3801 \glsnopostdotfalse
3802 }

stpunc@postdesc
3803 \newcommand*{\@glxtr@nopostpunc@postdesc}{%

restore@postpunc
3804 \newcommand*{\@glxtr@restore@postpunc}{%
3805   \def\@glxtr@nopostpunc@postdesc{%
3806     \@glxtr@org@postdescription
3807     \let\@glxtr@nopostpunc@postdesc\@empty
3808     \let\glxtrrestorepostpunc\@empty
3809   }%
3810 }

restorepostpunc Does nothing outside of glossary.
3811 \newcommand*{\glxtrrestorepostpunc}{%

\@printglossary Redefine.
3812 \renewcommand{\@printglossary}[2]{%
3813   \def\@glxtr@printglossopts{#1}%
3814   \@glxtr@orgprintglossary{#1}{#2}%
3815 }

Add a key that switches off the entry targets:
3816 \define@choicekey{printgloss}{target}
3817 [\@glxtr@printglossval\@glxtr@printglossnr]%
3818 {true,false}[true]%
3819 {%
3820   \ifcase\@glxtr@printglossnr
3821     \let\@glstarget\glsdohypertarget
3822   \else
3823     \let\@glstarget\@secondoftwo
3824   \fi
3825 }

hypernameprefix
3826 \newcommand{\@glxtrhypernameprefix}{%

New to v1.20:
3827 \define@key{printgloss}{targetnameprefix}{%
3828   \renewcommand{\@glxtrhypernameprefix}{#1}%
3829 }

```

`\glsdohypertarget` Redefine to insert `\@glxtrhypernameprefix` before the target name.

```
3830 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
3831 \renewcommand{\glsdohypertarget}[2]{%
3832   \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
3833 }
```

`@makeglossaries` For the benefit of `makeglossaries`

```
3834 \newcommand*{\glxtr@makeglossaries}[1]{}
```

`@glxtr@gettype` Get just the type.

```
3835 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
3836   \def\@glo@type{#2}%
3837 }
```

`@assign@sortkey` Assign the sort key.

```
3838 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
3839   \edef\@glo@type{\@glo@type}%
3840   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
3841   {%
3842     \@glo@no@assign@sortkey{#1}%
3843   }%
3844   {%
3845     \@glo@assign@sortkey{#1}%
3846   }%
3847 }%
```

Display number list for the regular version:

`splaynumberlist`

```
3848 \let\@glxtr@idx@displaynumberlist\glstdisplaynumberlist
```

Display number list for the “noidx” version:

`splaynumberlist`

```
3849 \newcommand*{\@glxtr@noidx@displaynumberlist}[1]{%
3850   \letcs{\@gls@loclist}{glo@\glstdetoklabel{#1}@loclist}%
3851   \ifdef\@gls@loclist
3852   {%
3853     \def\@gls@noidxloclist@sep{%
3854       \def\@gls@noidxloclist@sep{%
3855         \def\@gls@noidxloclist@sep{%
3856           \glsnumlistsep
3857         }%
3858         \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
3859       }%
3860     }%
3861     \def\@gls@noidxloclist@finalsep{}%
3862     \def\@gls@noidxloclist@prev{}%
3863     \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%

```

```

3864 \gls@noidxloclist@finalsep
3865 \gls@noidxloclist@prev
3866 }%
3867 {%

3868 \glsxtrundefrag
3869 \glsdoifexists{#1}%
3870 {%
3871 \GlossariesWarning{Missing location list for ‘#1’. Either
3872 a rerun is required or you haven’t referenced the entry.}%
3873 }%
3874 }%
3875 }%
3876

```

And for the number list loop:

@numberlistloop

```

3877 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
3878 \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
3879 \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
3880 \let\@gls@org@glsseeformat\glsseeformat
3881 \let\glsnoidxdisplayloc#2\relax
3882 \let\glsseeformat#3\relax
3883 \ifdef\@gls@loclist
3884 {%
3885 \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
3886 }%
3887 {%

3888 \glsxtrundefrag
3889 \glsdoifexists{#1}%
3890 {%
3891 \GlossariesWarning{Missing location list for ‘##1’. Either
3892 a rerun is required or you haven’t referenced the entry.}%
3893 }%
3894 }%
3895 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
3896 \let\glsseeformat\@gls@org@glsseeformat
3897 }%

```

Same for entry number list.

entrynumberlist

```

3898 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
3899 \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
3900 \ifdef\@gls@loclist
3901 {%
3902 \glsnoidxloclist{\@gls@loclist}%
3903 }%
3904 {%

```

```

3905 \glxstrundeftag
3906 \glsoifexists{#1}%
3907 {%
3908 \GlossariesWarning{Missing location list for ‘#1’. Either
3909 a rerun is required or you haven’t referenced the entry.}%
3910 }%
3911 }%
3912 }%

```

entrynumberlist

```

3913 \newcommand*{\@glxstr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

3914 \renewcommand*{\@glx@noidx@getgrouptitle}[2]{%
3915 \protected@edef\@glxstr@titlelabel{#1}%
3916 \ifdefvoid\@glxstr@titlelabel
3917 {}%
3918 {%
3919 \protected@edef\@glxstr@titlelabel{\csuse{glxstr@grouptitle@#1}}%
3920 }%
3921 \ifdefvoid{\@glxstr@titlelabel}%
3922 {%
3923 \DTLifint{#1}%
3924 {%
3925 \ifnum#1<256\relax
3926 \edef#2{\char#1\relax}%
3927 \else
3928 \edef#2{#1}%
3929 \fi
3930 }%
3931 {%
3932 \ifcsundef{#1groupname}%
3933 {\def#2{#1}}%
3934 {\letcs#2{#1groupname}}%
3935 }%
3936 }%
3937 {%
3938 \let#2\@glxstr@titlelabel
3939 }%
3940 }

```

g@getgrouptitle Save original definition of \@glx@getgrouptitle

```

3941 \let\glxstr@org@getgrouptitle\@glx@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.

```

3942 \newrobustcmd{\glxstrgetgrouptitle}[2]{%
3943 \protected@edef\@glxstr@titlelabel{glxstr@grouptitle@#1}%
3944 \@onelevel@sanitize\@glxstr@titlelabel

```

```

3945 \ifcsdef{\@glsxtr@titlelabel}
3946 {\letcs{#2}{\@glsxtr@titlelabel}}%
3947 {\glsxtr@org@getgrouptitle{#1}{#2}}%
3948 }
3949 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

trsetgrouptitle Sets the title for the given group label.

```

3950 \newcommand{\glsxtrsetgrouptitle}[2]{%
3951 \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
3952 \@onelevel@sanitize\@glsxtr@titlelabel
3953 \protected@csxdef{\@glsxtr@titlelabel}{#2}%
3954 }

```

alsetgrouptitle As above put only locally defines the title.

```

3955 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
3956 \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
3957 \@onelevel@sanitize\@glsxtr@titlelabel
3958 \protected@csedef{\@glsxtr@titlelabel}{#2}%
3959 }

```

\glsnavigation Redefine to use new user-level command.

```

3960 \renewcommand*{\glsnavigation}{%
3961 \def\@gls@between{}%
3962 \ifcsundef{\@gls@hypergroup@list@\@glo@type}%
3963 {%
3964 \def\@gls@list{}%
3965 }%
3966 {%
3967 \expandafter\let\expandafter\@gls@list
3968 \csname \@gls@hypergroup@list@\@glo@type\endcsname
3969 }%
3970 \@for\@gls@tmp:=\@gls@list\do{%
3971 \@gls@between
3972 \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
3973 \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
3974 \let\@gls@between\glshypernavsep
3975 }%
3976 }

```

@noidx@glossary

```

3977 \renewcommand*{\@print@noidx@glossary}{%
3978 \ifcsdef{\@glsref@\@glo@type}%
3979 {%
3980 \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
3981 {%
3982 \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
3983 }%
3984 {%
3985 \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%

```

```

3986 }%
3987 \glossarysection[\glossarytoctitle]{\glossarytitle}%
3988 \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

3989 \def\@gls@currentlettergroup{%
3990 \begin{theglossary}%
3991 \glossaryheader
3992 \glsresetentrylist
3993 \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}%
3994 \end{theglossary}%
3995 \glossarypostamble
3996 }%
3997 {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

3998 \glsxtrifemptyglossary{\@glo@type}%
3999 {}%
4000 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
4001 \@gls@noref@warn{\@glo@type}%
4002 }%
4003 }

```

noidxdisplayloc Patch to check for range formations.

```

4004 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4005 \setentrycounter[#1]{#2}%
4006 \@glsxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
4007 }

```

xtr@display@loc Patch to check for range formations.

```

4008 \def\@glsxtr@display@loc#1#2\end@glxtr@display@loc#3{%
4009 \ifx#1\relax
4010 \glsxtrdisplaystartloc{#2}{#3}%
4011 \else
4012 \ifx#1)\relax
4013 \glsxtrdisplayendloc{#2}{#3}%
4014 \else
4015 \glsxtrdisplaysingleloc{#1#2}{#3}%
4016 \fi
4017 \fi
4018 }

```

isplaysingleloc Single location.

```

4019 \newcommand*{\glsxtrdisplaysingleloc}[2]{%
4020 \csuse{#1}{#2}%
4021 }

```

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.

```
4022 \newcommand*{\glxtrdisplaystartloc}[2]{%
4023   \edef\glxtrlocrangefmt{#1}%
4024   \ifx\glxtrlocrangefmt\empty
4025     \def\glxtrlocrangefmt{\glsnnumberformat}%
4026   \fi
4027   \expandafter\glxtrdisplaysingleloc
4028   \expandafter{\glxtrlocrangefmt}{#2}%
4029 }
```

`\trdisplayendloc` End of a location range.

```
4030 \newcommand*{\glxtrdisplayendloc}[2]{%
4031   \edef\@glxtr@tmp{#1}%
4032   \ifdefempty{\@glxtr@tmp}{\def\@glxtr@tmp{\glsnnumberformat}}{}%
4033   \ifx\glxtrlocrangefmt\@glxtr@tmp
4034     \else
4035       \GlossariesExtraWarning{Mismatched end location range
4036         (start=\glxtrlocrangefmt, end=\@glxtr@tmp)}%
4037     \fi
4038     \expandafter\glxtrdisplayendlochook\expandafter{\@glxtr@tmp}{#2}%
4039     \expandafter\glxtrdisplaysingleloc
4040     \expandafter{\glxtrlocrangefmt}{#2}%
4041     \def\glxtrlocrangefmt{}%
4042 }
```

`\splayendlochook` Allow the user to hook into the end of range command.

```
4043 \newcommand*{\glxtrdisplayendlochook}[2]{}
```

`\sxtrlocrangefmt` Current range format. Empty if not in a range.

```
4044 \newcommand*{\glxtrlocrangefmt}{}%
```

`\setentrycounter` Adjust `\setentrycounter` to save the original prefix.

```
4045 \renewcommand*{\setentrycounter}[2][{}]{%
4046   \def\glxtrcounterprefix{#1}%
4047   \ifx\glxtrcounterprefix\@empty
4048     \def\@glo@counterprefix{.}%
4049   \else
4050     \def\@glo@counterprefix{.#1.}%
4051   \fi
4052   \def\glsetentrycounter{#2}%
4053 }
```

`\ls@removespaces` Redefine to allow adjustments to location hyperlink.

```
4054 \def\@gls@removespaces#1 #2\@nil{%
4055   \toks@=\expandafter{\the\toks@#1}%
4056 }
```

```

4056 \ifx\|#2\%
4057 \edef\x{\the\toks@}%
4058 \ifx\x\empty
4059 \else
    Expand location (just in case \toks@ is needed for something else).
4060 \expandafter\glxtrlocationhyperlink\expandafter
4061 \glstrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4062 \fi
4063 \else
4064 \@gls@ReturnAfterFi{%
4065 \@gls@removespaces#2\@nil
4066 }%
4067 \fi
4068 }

```

cationhyperlink

```

4069 \newcommand*\glxtrlocationhyperlink}[3]{%
4070 \ifdefvoid\glxtrsupplocationurl
4071 {%
4072 \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
4073 }%
4074 {%
4075 \hyperref{\glxtrsupplocationurl}{#1#2#3}{#3}%
4076 }%
4077 }

```

supphypernumber

```

4078 \newcommand*\glxtrsupphypernumber}[1]{%
4079 {%
4080 \glshasattribute{\glscurrententrylabel}{externallocation}%
4081 {%
4082 \def\glxtrsupplocationurl{%
4083 \glsggetattribute{\glscurrententrylabel}{externallocation}}%
4084 }%
4085 {%
4086 \def\glxtrsupplocationurl{}%
4087 }%
4088 \glshypernumber{#1}%
4089 }%
4090 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

4091 \renewcommand\@print@glossary{%
4092 \makeatletter
4093 \@input{\jobname.\csname @glotype@\@glo@type @in\endcsname}%

```

```

4094 \IfFileExists{\jobname.\csname @glo@type @in\endcsname}%
4095 {}%
4096 {\glstrNoGlossaryWarning{\@glo@type}}%
4097 \ifglxindy
4098   \ifcsundef{@xdy@\@glo@type @language}%
4099   {%
4100     \edef\@do@auxoutstuff{%
4101       \noexpand\AtEndDocument{%
4102         \noexpand\immediate\noexpand\write\@auxout{%
4103           \string\providecommand\string\@xdylanguage[2]{}}%
4104         \noexpand\immediate\noexpand\write\@auxout{%
4105           \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4106       }%
4107     }%
4108   }%
4109   {%
4110     \edef\@do@auxoutstuff{%
4111       \noexpand\AtEndDocument{%
4112         \noexpand\immediate\noexpand\write\@auxout{%
4113           \string\providecommand\string\@xdylanguage[2]{}}%
4114         \noexpand\immediate\noexpand\write\@auxout{%
4115           \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4116             @language\endcsname}}%
4117       }%
4118     }%
4119   }%
4120   \@do@auxoutstuff
4121   \edef\@do@auxoutstuff{%
4122     \noexpand\AtEndDocument{%
4123       \noexpand\immediate\noexpand\write\@auxout{%
4124         \string\providecommand\string\@gls@codepage[2]{}}%
4125       \noexpand\immediate\noexpand\write\@auxout{%
4126         \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
4127     }%
4128   }%
4129   \@do@auxoutstuff
4130 \fi
4131 \renewcommand*{\@warn@nomakeglossaries}{%
4132   \GlossariesWarningNoLine{\string\makeglossaries\space
4133     hasn't been used,^^Jthe glossaries will not be updated}%
4134 }%
4135 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`oGlsWarningHead` Header message.

```

4136 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4137   This document is incomplete. The external file associated with
4138   the glossary '#1' (which should be called \texttt{#2})
4139   hasn't been created.%

```

```

4140 }

arningEmptyStart  No entries have been added to the glossary.
4141 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4142   This has probably happened because there are no entries defined
4143   in this glossary.%
4144 }

arningEmptyMain  The default “main” glossary is empty.
4145 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4146   If you don’t want this glossary,
4147   add \texttt{nomain} to your package option list when you load
4148   \texttt{glossaries-extra.sty}. For example:%
4149 }

ingEmptyNotMain  A glossary that isn’t the default “main” glossary is empty.
4150 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4151   Did you forget to use \texttt{type=#1} when you defined your
4152   entries? If you tried to load entries into this glossary with
4153   \texttt{\string\loadglsentries} did you remember to use
4154   \texttt{[#1]} as the optional argument? If you did, check that
4155   the definitions in the file you loaded all had the type set
4156   to \texttt{\string\glsdefaulttype}.%
4157 }

arningCheckFile  Advisory message to check the file contents.
4158 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4159   Check the contents of the file \texttt{#1}. If
4160   it’s empty, that means you haven’t indexed any of your entries in this
4161   glossary (using commands like \texttt{\string\gls} or
4162   \texttt{\string\glsadd}) so this list can’t be generated.
4163   If the file isn’t empty, the document build process hasn’t been
4164   completed.%
4165 }

WarningAutoMake  Message when automake option has been used.
4166 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4167   You may need to rerun \LaTeX. If you already have, it may be that
4168   \TeX’s shell escape doesn’t allow you to run
4169   \ifglxindy xindy\else makeindex\fi. Check the
4170   transcript file \texttt{\jobname.log}. If the shell escape is
4171   disabled, try one of the following:
4172
4173   \begin{itemize}
4174     \item Run the external (Lua) application:
4175
4176       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4177
4178     \item Run the external (Perl) application:

```

```

4179
4180     \texttt{makeglossaries \string"\jobname\string"}
4181 \end{itemize}
4182
4183 Then rerun \LaTeX\ on this document.
4184 \GlossariesExtraWarning{Rerun required to build the
4185 glossary ‘#1’ or check TeX’s shell escape allows
4186 you to run \ifglxindy xindy\else makeindex\fi}%
4187 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

4188 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
4189   You need to either replace \texttt{\string\makenoidxglossaries}
4190   with \texttt{\string\makeglossaries} or replace
4191   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
4192   \texttt{\string\printnoidxglossary}
4193   (or \texttt{\string\printnoidxglossaries}) and then rebuild
4194   this document.%
4195 }

```

WarningBuildInfo Build advice.

```

4196 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4197   Try one of the following:
4198   \begin{itemize}
4199     \item Add \texttt{automake} to your package option list when you load
4200       \texttt{glossaries-extra.sty}. For example:
4201
4202       \texttt{\string\usepackage[automake]%
4203         \glsopenbrace glossaries-extra\glsclosebrace}
4204
4205     \item Run the external (Lua) application:
4206
4207       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4208
4209     \item Run the external (Perl) application:
4210
4211       \texttt{makeglossaries \string"\jobname\string"}
4212   \end{itemize}
4213
4214   Then rerun \LaTeX\ on this document.%
4215 }

```

oGlsWarningTail Final paragraph.

```

4216 \newcommand{\GlsXtrNoGlsWarningTail}{%
4217   This message will be removed once the problem has been fixed.%
4218 }

```

GlsWarningNoOut No out file created. Build advice.

```

4219 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4220   The file \texttt{#1} doesn't exist. This most likely means you haven't used
4221   \texttt{\string\makeglossaries} or you have used
4222   \texttt{\string\nofiles}. If this is just a draft version of the
4223   document, you can suppress this message using the
4224   \texttt{nomissingglstext} package option.%
4225 }

```

glossarywarning

```

4226 \newcommand*{\@glsxtr@defaultnoglossarywarning}[1]{%
4227   \glossarysection[\glossarytoctitle]{\glossarytitle}
4228   \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
4229   \par
4230   \glsxtrifemptyglossary{#1}%
4231   {%
4232     \GlsXtrNoGlsWarningEmptyStart\space
4233     \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4234     \medskip
4235     \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
4236       \glsopenbrace glossaries-extra\glsclosebrace}
4237     \medskip
4238     }%
4239     {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4240   }%
4241   {%
4242     \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4243     {%
4244       \GlsXtrNoGlsWarningCheckFile
4245       {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4246
4247       \ifglsautomake
4248
4249       \GlsXtrNoGlsWarningAutoMake{#1}
4250
4251     \else
4252
4253       \ifthenelse{\equal{#1}{main}}{%
4254       {%
4255         \GlsXtrNoGlsWarningEmptyMain\par
4256         \medskip
4257         \noindent\texttt{\string\usepackage[nomain]%
4258           \glsopenbrace glossaries-extra\glsclosebrace}
4259         \medskip
4260       }%
4261       }%
4262
4263       \ifdefequal\makeglossaries\@no@makeglossaries
4264       {%
4265         \GlsXtrNoGlsWarningMisMatch

```

```

4266     }%
4267     {%
4268         \GlsXtrNoGlsWarningBuildInfo
4269     }%
4270     \fi
4271 }%
4272 {%
4273     \GlsXtrNoGlsWarningNoOut
4274     {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
4275 }%
4276 }%
4277 \par
4278 \GlsXtrNoGlsWarningTail
4279 }

```

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.

```

4280 \newcommand*{\glxtrresourcefile}[2] [] {%

```

The record option can't be set after this command.

```

4281     \disable@keys{glossaries-extra.sty}{record}%
4282     \glxtr@writefields
4283     \protected@write\@auxout{\glxtrresourceinit}{\string\glxtr@resource{#1}{#2}}%
4284     \let\@glxtr@org@see@noindex\@gls@see@noindex
4285     \let\@gls@see@noindex\relax
4286     \IfFileExists{#2.glstex}%
4287     {%

```

Can't scope \@input so save and restore the category code of @ to allow for internal commands in the location list.

```

4288     \edef\@bibgls@restoreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
4289     \makeatletter
4290     \@input{#2.glstex}%
4291     \@bibgls@restoreat
4292 }%
4293 {%
4294     \GlossariesExtraWarning{No file '#2.glstex'}%
4295 }%
4296     \let\@gls@see@noindex\@glxtr@org@see@noindex
4297 }
4298 \@onlypreamble\glxtrresourcefile

```

xtresourceinit Code used during the protected write operation.

```

4299 \newcommand*{\glxtrresourceinit}{%

```

trresourcecount

```

4300 \newcount\glxtrresourcecount

```

trLoadResources Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.

```
4301 \newcommand*{\GlsXtrLoadResources}[1][\relax]{%
4302   \ifnum\glxtrresourcecount=0\relax
4303     \glxtrresourcefile[#1]{\jobname}%
4304   \else
4305     \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
4306   \fi
4307   \advance\glxtrresourcecount by 1\relax
4308 }
```

glxtr@resource

```
4309 \newcommand*{\glxtr@resource}[2]{}
```

\glxtr@fields

```
4310 \newcommand*{\glxtr@fields}[1]{}
```

xtr@texencoding

```
4311 \newcommand*{\glxtr@texencoding}[1]{}
```

\glxtr@langtag

```
4312 \newcommand*{\glxtr@langtag}[1]{}
```

@pluralsuffixes

```
4313 \newcommand*{\glxtr@pluralsuffixes}[4]{}
```

tr@shortcutsval

```
4314 \newcommand*{\glxtr@shortcutsval}[1]{}
```

sxtr@linkprefix

```
4315 \newcommand*{\glxtr@linkprefix}[1]{}
```

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```
4316 \newcommand*{\glxtr@writefields}{%

4317   \protected@write\@auxout{%
4318     {\string\providecommand*{\string\glxtr@fields}[1]{}}%
4319   \protected@write\@auxout{%
4320     {\string\providecommand*{\string\glxtr@resource}[2]{}}%
4321   \protected@write\@auxout{%
4322     {\string\providecommand*{\string\glxtr@pluralsuffixes}[4]{}}%
4323   \protected@write\@auxout{%
4324     {\string\providecommand*{\string\glxtr@shortcutsval}[1]{}}%
4325   \protected@write\@auxout{%
4326     {\string\providecommand*{\string\glxtr@linkprefix}[1]{}}%
4327   \protected@write\@auxout{{\string\glxtr@fields{\@gls@keymap}}}%

4328   \protected@write\@auxout{%
4329     {\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`.

```

4330 \ifdef\CurrentTrackedLanguageTag
4331 {%
4332     \protected@write\@auxout{}\string\glxtr@langtag{\CurrentTrackedLanguageTag}}%
4333 }%
4334 {%
4335     \protected@write\@auxout{}\string\glxtr@pluralsuffixes
4336     {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
4337     {\glxtrabbrvpluralsuffix}}%
4338 \ifdef\inputencodingname
4339 {%
4340     \protected@write\@auxout{}\string\glxtr@texencoding{\inputencodingname}}%
4341 }%
4342 {%
4343 
```

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.)

```

4344     \@ifpackageloaded{fontspec}%
4345     {\protected@write\@auxout{}\string\glxtr@texencoding{utf8}}%
4346     }%
4347 }%
4348 \protected@write\@auxout{}\string\glxtr@shortcutsval{\glxtr@shortcutsval}}%

```

Prefix deferred until the beginning of the document in case it's redefined later in the preamble. This is picked up by `bib2gls` when the external option is used.

```

4349 \AtBeginDocument
4350 {\protected@write\@auxout{}\string\glxtr@linkprefix{\glolinkprefix}}%
4351 \let\glxtr@writefields\relax

```

If the `automake` option is on, try running `bib2gls` if the aux file exists. The double-quotes around `\jobname` have been removed (v1.19) since `\jobname` will include double-quotes if the file name has spaces.

```

4352 \ifglautomake
4353 \IfFileExists{\jobname.aux}%
4354 {\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`.

```

4355 \ifx\gls@doautomake\gls@doautomake@err
4356 \let\gls@doautomake\relax
4357 \fi
4358 \fi
4359 }

```

`do@automake@err`

```

4360 \newcommand*{\@gls@doautomake@err}{%
4361 \PackageError{glossaries}{You must use

```

```

4362 \string\makeglossaries\space with automake=true}
4363 {%
4364     Either remove the automake=true setting or
4365     add \string\makeglossaries\space to your document preamble.%
4366 }%
4367 }

```

Allow locations specific to a particular counter to be recorded.

\glxtr@record

```

4368 \newcommand*{\glxtr@record}[5]{}

```

r@counterrecord Aux file command.

```

4369 \newcommand*{\glxtr@counterrecord}[3]{%
4370   \glxtrfieldlistgadd{#1}{record.#2}{#3}%
4371 }

```

unterrecordhook Hook used by \@glxtr@dorecord.

```

4372 \newcommand*{\@glxtr@counterrecordhook}{}

```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```

4373 \newcommand*{\GlsXtrRecordCounter}[1]{%
4374   \@glxtr@recordcounter{#1}%
4375 }
4376 \@onlypreamble\GlsXtrRecordCounter

```

docounterrecord

```

4377 \newcommand*{\@glxtr@docounterrecord}[1]{%
4378   \protected@write\@auxout{}\string\glxtr@counterrecord
4379     {\@glx@label}{#1}{\csuse{the#1}}}%
4380 }

```

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.

```

4381 \newcommand*{\glxtrglossentry}[1]{%
4382   \glxtrtitleorpdforheading
4383   {\@glxtrglossentry{#1}}%
4384   {\glsentryname{#1}}%
4385   {\glxtrheadname{#1}}%
4386 }

```

lsxtrglossentry Another test is needed in case \@glxtrglossentry has been written to the table of contents.

```

4387 \newrobustcmd*{\@glxtrglossentry}[1]{%

```

```

4388 \glstrtitleorpdforheading
4389 {%
4390   \glsoifexists{#1}%
4391   {%
4392     \beginngroup
4393     \edef\glscurrententrylabel{\glsetoklabel{#1}}%
4394     \edef\currentglossary{\glsetentrytype{\glscurrententrylabel}}%
4395     \ifglshasparent{#1}%
4396     {\glssubentryitem{#1}}%
4397     {\glsetentryitem{#1}}%
4398     \glstarget{#1}{\glossentryname{#1}}%
4399   \endgroup
4400   }%
4401 }%
4402 {\glsetentryname{#1}}%
4403 {\glstrheadname{#1}}%
4404 }

```

`glossentryother` As `\glstrglossentry` 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.

```

4405 \newcommand*{\glstrglossentryother}[3]{%
4406   \ifstrempy{#1}%
4407   {%
4408     \ifcsdef{glstrhead#3}%
4409     {%
4410       \glstrtitleorpdforheading
4411       {\@glstrglossentryother{#2}{#3}{#1}}%
4412       {\@gls@entry@field{#2}{#3}}%
4413       {\csuse{glstrhead#3}{#2}}%
4414     }%
4415     {%
4416       \glstrtitleorpdforheading
4417       {\@glstrglossentryother{#2}{#3}{#1}}%
4418       {\@gls@entry@field{#2}{#3}}%
4419       {\@gls@entry@field{NoCaseChange{#2}}{#3}}%
4420     }%
4421   }%
4422   {%
4423     \glstrtitleorpdforheading
4424     {\@glstrglossentryother{#2}{#3}{#1}}%
4425     {\@gls@entry@field{#2}{#3}}%
4426     {#1}%
4427   }%
4428 }

```

`glossentryother` As `\@glstrglossentry` but uses a different field.

```

4429 \newrobustcmd*{\@glstrglossentryother}[3]{%

```

```

4430 \glstrtitleorpdforheading
4431 {%
4432   \glsdoifexists{#1}%
4433   {%
4434     \begingroup
4435     \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
4436     \edef\currentglossary{\glssentrytype{\glscurrententrylabel}}%
4437     \ifglshasparent{#1}%
4438     {\glssubentryitem{#1}}%
4439     {\glssentryitem{#1}}%
4440     \glstarget{#1}{\glossentrynameother{#1}{#2}}%
4441   \endgroup
4442   }%
4443 }%
4444 {\@gls@entry@field{#1}{#2}}%
4445 {#3}%
4446 }

```

`\printunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```

4447 \newcommand*{\printunsrtglossary}{%
4448   \@ifstar\s@printunsrtglossary\@printunsrtglossary
4449 }

```

`\printunsrtglossary` Unstarred version.

```

4450 \newcommand*{\@printunsrtglossary}[1][\]{%
4451   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4452 }

```

`\printunsrtglossary` Starred version.

```

4453 \newcommand*{\s@printunsrtglossary}[2][\]{%
4454   \begingroup
4455   #2%
4456   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
4457   \endgroup
4458 }

```

`\printunsrtglossaries` Similar to `\printnoidxglossaries` but it displays all entries defined for the given glossary without sorting.

```

4459 \newcommand*{\printunsrtglossaries}{%
4460   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
4461 }

```

`\@unsrt@glossary`

```

4462 \newcommand*{\@print@unsrt@glossary}{%
4463   \glossarysection[\glossarytoctitle]{\glossarytitle}%
4464   \glossarypreamble

```

check for empty list

```

4465 \glxtrifemptyglossary{\@glo@type}%
4466 {%
4467   \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
4468 }%
4469 {%

4470   \key@ifundefined{glossentry}{group}%
4471   {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
4472   {\let\@gls@getgrouptitle\@glxtr@unsrt@getgrouptitle}%
4473   \def\@gls@currentlettergroup{}}%

```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```

4474 \def\@glxtr@doglossary{%
4475   \begin{theglossary}%
4476   \glossaryheader
4477   \glsresetentrylist
4478 }%
4479 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
4480   :\expandafter=\csname glist@\@glo@type\endcsname\do{%
4481   \ifdefempty{\glscurrententrylabel}
4482   }%
4483   {%

```

Provide a hook (for example to measure width).

```

4484   \let\glxtr@process\@firstofone
4485   \let\printunsrtglossaryskipentry
4486       \@glxtr@printunsrtglossaryskipentry
4487   \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%

```

Don't check group for child entries.

```

4488   \glxtr@process
4489   {%
4490     \ifglshasparent{\glscurrententrylabel}{}%
4491     {%
4492       \@glxtr@checkgroup\glscurrententrylabel
4493       \expandafter\appto\expandafter\@glxtr@doglossary\expandafter
4494         {\@glxtr@groupheading}%
4495     }%
4496     \eappto\@glxtr@doglossary{%
4497       \noexpand\@printunsrtglossary@handler{\glscurrententrylabel}}%
4498   }%
4499 }%
4500 }%
4501 \appto\@glxtr@doglossary{\end{theglossary}}%
4502 \printunsrtglossarypredoglossary
4503 \@glxtr@doglossary
4504 }%
4505 \glossarypostamble
4506 }

```

entryprocesshook

```
4507 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{}
```

entryprocesshook

```
4508 \newcommand*{\printunsrtglossaryskipentry}{%
4509   \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
4510 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
4511 }
```

entryprocesshook

```
4512 \newcommand*{\@glxstr@printunsrtglossaryskipentry}{%
4513   \let\glxstr@process\@gobble
4514 }
```

entrypredoglossary

```
4515 \newcommand*{\printunsrtglossarypredoglossary}{}%
```

glossary@handler

```
4516 \newcommand{\@printunsrt@glossary@handler}[1]{%
4517   \xdef\glscurrententrylabel{#1}%
4518   \printunsrtglossaryhandler\glscurrententrylabel
4519 }
```

glossaryhandler

```
4520 \newcommand{\printunsrtglossaryhandler}[1]{%
4521   \glxstrunsrtdo{#1}%
4522 }
```

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.

```
4523 \newrobustcmd*{\glxstriflabelinlist}[4]{%
4524   \protected@edef\@glxstr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
4525   \@glxstr@doiflabelinlist{#3}{#4}%
4526 }
```

unsrtglossaryunit

```
4527 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
4528   \s@printunsrtglossary[type=\glstypedefaulttype,#1]{%
4529     \printunsrtglossaryunitsetup{#2}%
4530   }%
4531 }
```

glossaryunitsetup

```
4532 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
4533   \renewcommand{\printunsrtglossaryhandler}[1]{%
4534     \glxstrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
```

```

4535   {\glxtrunsrtdo{##1}}%
4536   {}%
4537 }%

```

Only the target names should have the prefixes adjusted as \gls etc need the original \glolinkprefix. The \@gobble part discards \glolinkprefix.

```

4538 \ifcsundef{theH#1}%
4539 {%
4540   \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
4541 }%
4542 {%
4543   \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
4544 }%
4545 \renewcommand*{\glossarysection}[2] [] {}%
4546 \appto\glossarypostamble{\glspar\medskip\glspar}%
4547 }

```

srtglossaryunit

```

4548 \newcommand{\print@noop@unsrtglossaryunit}[2] [] {}%
4549 \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
4550   requires the record=only or record=alsoindex package option}{}%
4551 }

```

t@getgrouptitle

```

4552 \newrobustcmd*{\@glxtr@unsrt@getgrouptitle}[2] {}%
4553 \protected@edef\@glxtr@titlelabel{glxtr@grouptitle@#1}%
4554 \@onelevel@sanitize\@glxtr@titlelabel
4555 \ifcsdef{\@glxtr@titlelabel}
4556 {\letcs{#2}{\@glxtr@titlelabel}}%
4557 {\def#2{#1}}%
4558 }

```

\glxtrunsrtdo Provide a user-level call to \@glxtr@noidx@do to make it easier to define a new handler.

```

4559 \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).

```

4560 \newcommand*{\lsxtrgroupfield}{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.

```

4561 \newcommand*{\@glsxtr@checkgroup}[1]{%
4562   \def\@glsxtr@groupheading{}%
4563   \key@ifundefined{glossentry}{group}%
4564   {%
4565     \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
4566     \expandafter\glo@grabfirst\@gls@sort{}\@nil
4567   }%
4568   {%

4569     \protected@edef\@glo@thislettergrp{%
4570       \csuse{glo@\glsdetoklabel{#1}@\glsxtrgroupfield}}%
4571   }%
4572   \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
4573   {}%
4574   {%
4575     \ifdefempty{\@gls@currentlettergroup}{}%
4576     {\def\@glsxtr@groupheading{\gls@groupskip}}%
4577     \eappto\@glsxtr@groupheading{%
4578       \noexpand\gls@groupheading{\expandonce\@glo@thislettergrp}%
4579     }%
4580   }%
4581   \let\@gls@currentlettergroup\@glo@thislettergrp
4582 }

```

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.

```

4583 \newcommand{\@glsxtr@noidx@do}[1]{%
4584   \ifglstryexists{#1}%
4585   {%
4586     \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
4587     \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@location}%
4588     \ifglshasparent{#1}%
4589     {%
4590       \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
4591       \ifdefvoid{\@gls@location}%
4592       {%
4593         \ifdefvoid{\@gls@loclist}%
4594         {%
4595           \subglossentry{\gls@level}{#1}{}%
4596         }%
4597       }%
4598       \subglossentry{\gls@level}{#1}%
4599     }%
4600     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4601   }%
4602 }%
4603 }%
4604 {%
4605   \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
4606 }%

```

```

4607 }%
4608 {%

4609 \ifdefvoid{\@gls@location}%
4610 {%
4611 \ifdefvoid{\@gls@loclist}
4612 {%
4613 \glossentry{#1}{}%
4614 }%
4615 {%
4616 \glossentry{#1}%
4617 {%
4618 \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
4619 }%
4620 }%
4621 }%
4622 {%
4623 \glossentry{#1}%
4624 {%
4625 \glossaryentrynumbers{\@gls@location}%
4626 }%
4627 }%
4628 }%
4629 }%
4630 {}%
4631 }

```

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`

```
4632 \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>}
```

```

4633 \newcommand*{\@glsxtrnewgls}[4]{%
4634 \ifdef{#3}%
4635 {%
4636 \PackageError{glossaries-extra}{Command \string#3\space already
4637 defined}{}%
4638 }%
4639 {%

```

```

4640 \ifcsdef{@#4like@#2}%
4641 {%
4642 \advance\@glxstrnewgls@inner by \@ne
4643 \def\@glxstrnewgls@innercsname{@#4like\number\@glxstrnewgls@inner @#2}%
4644 }%
4645 {\def\@glxstrnewgls@innercsname{@#4like@#2}}%
4646 \expandafter\newrobustcmd\expandafter*\expandafter
4647 #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glxstrnewgls@innercsname\endcsname}%
4648 \ifstrempy{#1}%
4649 {%
4650 \expandafter\newcommand\expandafter*\csname\@glxstrnewgls@innercsname\endcsname[2][ ]{%
4651 \new@ifnextchar[%
4652 {\csname @#4@\endcsname{##1}{#2##2}}%
4653 {\csname @#4@\endcsname{##1}{#2##2}[]}%
4654 }%
4655 }%
4656 {%
4657 \expandafter\newcommand\expandafter*\csname\@glxstrnewgls@innercsname\endcsname[2][ ]{%
4658 \new@ifnextchar[%
4659 {\csname @#4@\endcsname{#1,##1}{#2##2}}%
4660 {\csname @#4@\endcsname{#1,##1}{#2##2}[]}%
4661 }%
4662 }%
4663 }%
4664 }

```

`\glxstrnewgls` `\glxstrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```

4665 \newrobustcmd*{\glxstrnewgls}[3][ ]{%
4666 \@glxstrnewgls{#1}{#2}{#3}{gls}%
4667 }

```

`\glxstrnewglslike` Provide a way to conveniently define commands that behave like `\gls`, `\glsp1`, `\Gls` and `\Glsp1` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

4668 \newrobustcmd*{\glxstrnewglslike}[6][ ]{%
4669 \@glxstrnewgls{#1}{#2}{#3}{gls}%
4670 \@glxstrnewgls{#1}{#2}{#4}{glsp1}%
4671 \@glxstrnewgls{#1}{#2}{#5}{Gls}%
4672 \@glxstrnewgls{#1}{#2}{#6}{Glsp1}%
4673 }

```

`\glxstrnewGLSlike` Provide a way to conveniently define commands that behave like `\GLS`, `\GLSp1` with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

4674 \newrobustcmd*{\glxstrnewGLSlike}[4][ ]{%

```

```

4675 \@glxstrnewgls{#1}{#2}{#3}{GLS}%
4676 \@glxstrnewgls{#1}{#2}{#4}{GLSpl}%
4677 }

```

\glxstrnewrgls As \glxstrnewgls but for \rgls.

```

4678 \newrobustcmd*{\glxstrnewrgls}[3][]{%
4679 \@glxstrnewgls{#1}{#2}{#3}{rgls}%
4680 }

```

sxtrnewrglslike As \glxstrnewrglslike but for \rgls etc.

```

4681 \newrobustcmd*{\glxstrnewrglslike}[6][]{%
4682 \@glxstrnewgls{#1}{#2}{#3}{rgls}%
4683 \@glxstrnewgls{#1}{#2}{#4}{rglspl}%
4684 \@glxstrnewgls{#1}{#2}{#5}{rGls}%
4685 \@glxstrnewgls{#1}{#2}{#6}{rGlspl}%
4686 }

```

sxtrnewrGLSlike As \glxstrnewrGLSlike but for \rGLS etc.

```

4687 \newrobustcmd*{\glxstrnewrGLSlike}[4][]{%
4688 \@glxstrnewgls{#1}{#2}{#3}{rGLS}%
4689 \@glxstrnewgls{#1}{#2}{#4}{rGLSpl}%
4690 }

```

Provide easy access to record count fields.

totalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```

4691 \newcommand*{\GlsXtrTotalRecordCount}[1]{%
4692 \ifcsdef{glo@\glstetoklabel{#1}@recordcount}%
4693 {\csname glo@\glstetoklabel{#1}@recordcount\endcsname}%
4694 {0}%
4695 }

```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```

4696 \newcommand*{\GlsXtrRecordCount}[2]{%
4697 \ifcsdef{glo@\glstetoklabel{#1}@recordcount.#2}%
4698 {\csname glo@\glstetoklabel{#1}@recordcount.#2\endcsname}%
4699 {0}%
4700 }

```

tionRecordCount Access record count for a particular counter and location. The first argument is the label. The second argument is the counter name. The third argument is the location. This command shouldn't be used if the location doesn't fully expand unless \glxstrdetoklocation can be set to something sensible.

```

4701 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
4702 \ifcsdef{glo@\glstetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}}%
4703 {\csname glo@\glstetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}\endcsname}%
4704 {0}%
4705 }

```

trdetoklocation

```
4706 \newcommand*{\glxtrdetoklocation}[1]{#1}
```

ablerecordcount

```
4707 \newcommand*{\glxtrenablerecordcount}{%
4708   \renewcommand*{\gls}{\rgls}%
4709   \renewcommand*{\Gls}{\rGls}%
4710   \renewcommand*{\glspl}{\rglspl}%
4711   \renewcommand*{\Glspl}{\rGlspl}%
4712   \renewcommand*{\GLS}{\rGLS}%
4713   \renewcommand*{\GLSpl}{\rGLSpl}%
4714 }
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
4715 \newcommand*{\glxtrrecordtriggervalue}[1]{%
4716   \GlsXtrTotalRecordCount{#1}%
4717 }
```

dCountAttribute

```
4718 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
4719   \@for\@glxtr@cat:=#1\do
4720   {%
4721     \ifdefempty{\@glxtr@cat}{}%
4722     {%
4723       \glssetcategoryattribute{\@glxtr@cat}{recordcount}{#2}%
4724     }%
4725   }%
4726 }
```

trifrecordtrigger

<code>\glxtrifrecordtrigger{<label>}{<trigger format>}{<normal>}</code>

```
4727 \newcommand*{\glxtrifrecordtrigger}[3]{%
4728   \glshasattribute{#1}{recordcount}%
4729   {%
4730     \ifnum\glxtrrecordtriggervalue{#1}>\glsggetattribute{#1}{recordcount}\relax
4731     #3%
4732   \else
4733     #2%
4734   \fi
4735   }%
4736   {#3}%
4737 }
```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```
4738 \newcommand*{\@glxtr@rglstrigger@record}[3]{%
```

```

4739 \edef\glslabel{\glsdetoklabel{#2}}%
4740 \let\@gls@link@label\glslabel
4741 \def\@glsxtr@thevalue{%
4742 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
4743 \def\@glsnumberformat{glstriggerrecordformat}%
4744 \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
4745 \edef\@gls@type{\csname glo@\glslabel @type\endcsname}%
4746 \def\@glsxtr@thevalue{%
4747 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
4748 \glsxtrinitwrgloss
4749 \glslinkpresetkeys
4750 \setkeys{glslink}{#1}%
4751 \glslinkpostsetkeys
4752 \ifdefempty{\@glsxtr@thevalue}%
4753 {%
4754 \@gls@saveentrycounter
4755 }%
4756 {%
4757 \let\theglsentrycounter\@glsxtr@thevalue
4758 \def\theHglentrycounter{\@glsxtr@theHvalue}%
4759 }%
4760 \ifglsxtrinitwrglossbefore
4761 \do@wrglossary{#2}%
4762 \fi
4763 #3%
4764 \ifglsxtrinitwrglossbefore
4765 \else
4766 \do@wrglossary{#2}%
4767 \fi
4768 \ifKV@glslink@local
4769 \glslocalunset{#2}%
4770 \else
4771 \glsunset{#2}%
4772 \fi
4773 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

4774 \newcommand*{\glstriggerrecordformat}[1]{

```

```

\rgls

```

```

4775 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

```

\@rgls

```

```

4776 \newcommand*{\@rgls}[2][{}]{%
4777 \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}}[{}]}%
4778 }

```

```

\@rgls@

```

```

4779 \def\@rgls@#1#2[#3]{%
4780   \glstrifrecordtrigger{#2}%
4781   {%
4782     \glstr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
4783   }%
4784   {%
4785     \@rgls@{#1}{#2}[#3]%
4786   }%
4787 }%

```

\rglspl

```

4788 \newrobustcmd*{\rglspl}{\@rgls@hyp@opt\@rglspl}

```

\@rglspl

```

4789 \newcommand*{\@rglspl}[2][{}]{%
4790   \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
4791 }

```

\@rglspl@

```

4792 \def\@rglspl@#1#2[#3]{%
4793   \glstrifrecordtrigger{#2}%
4794   {%
4795     \glstr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
4796   }%
4797   {%
4798     \@rglspl@{#1}{#2}[#3]%
4799   }%
4800 }%

```

\rGls

```

4801 \newrobustcmd*{\rGls}{\@rgls@hyp@opt\@rGls}

```

\@rGls

```

4802 \newcommand*{\@rGls}[2][{}]{%
4803   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
4804 }

```

\@rGls@

```

4805 \def\@rGls@#1#2[#3]{%
4806   \glstrifrecordtrigger{#2}%
4807   {%
4808     \glstr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
4809   }%
4810   {%
4811     \@Gls@{#1}{#2}[#3]%
4812   }%
4813 }%

```

```

\rGlspl
4814 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\rGlspl}

\@rGlspl
4815 \newcommand*{\@rGlspl}[2] [] {%
4816   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2} [] }%
4817 }

\@rGlspl@
4818 \def\@rGlspl@#1#2[#3]{%
4819   \glsxtrifrecordtrigger{#2}%
4820   {%
4821     \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
4822   }%
4823   {%
4824     \@Glspl@{#1}{#2}[#3]%
4825   }%
4826 }%

\rGLS
4827 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\rGLS}

\@rGLS
4828 \newcommand*{\@rGLS}[2] [] {%
4829   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2} [] }%
4830 }

\@rGLS@
4831 \def\@rGLS@#1#2[#3]{%
4832   \glsxtrifrecordtrigger{#2}%
4833   {%
4834     \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%
4835   }%
4836   {%
4837     \@GLS@{#1}{#2}[#3]%
4838   }%
4839 }%

\rGLSpl
4840 \newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\rGLSpl}

\@rGLSpl
4841 \newcommand*{\@rGLSpl}[2] [] {%
4842   \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2} [] }%
4843 }

```

```

\@rGLSpl@
4844 \def \@rGLSpl@#1#2[#3]{%
4845   \glxtrifrecordtrigger{#2}%
4846   {%
4847     \@glxtr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
4848   }%
4849   {%
4850     \@GLSpl@{#1}{#2}[#3]%
4851   }%
4852 }%

\rglsformat
4853 \newcommand*{\rglsformat}[2]{%
4854   \glsifregular{#1}
4855   {\glsentryfirst{#1}}%
4856   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
4857 }

\rglsplformat
4858 \newcommand*{\rglsplformat}[2]{%
4859   \glsifregular{#1}
4860   {\glsentryfirstplural{#1}}%
4861   {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
4862 }

\rGlsformat
4863 \newcommand*{\rGlsformat}[2]{%
4864   \glsifregular{#1}
4865   {\Glsentryfirst{#1}}%
4866   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
4867 }

\rGlsplformat
4868 \newcommand*{\rGlsplformat}[2]{%
4869   \glsifregular{#1}
4870   {\Glsentryfirstplural{#1}}%
4871   {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
4872 }

\rGLSformat
4873 \newcommand*{\rGLSformat}[2]{%
4874   \expandafter\mfirstucMakeUppercase\expandafter{\rglsformat{#1}{#2}}%
4875 }

\rGLSplformat
4876 \newcommand*{\rGLSplformat}[2]{%
4877   \expandafter\mfirstucMakeUppercase\expandafter{\rglsplformat{#1}{#2}}%
4878 }

```

1.4 Link Counting

This is different to the entry counting provided by the base package (which counts the number of times the first use flag is unset). Instead, this method hooks into `\@gls@link` (through `\glsxtr@inc@linkcount`) to increment an associated counter. To preserve resources, the counter is only defined if it needs to be incremented. This method is independent of the presence of hyperlinks. (The “link” part of the name refers to `\@gls@link` not `\hyperlink`.)

`\@inc@linkcount` This performs the actual incrementing and counter definition. The counter is given by `\c@glsxtr@linkcount@<label>` where `<label>` is the entry’s label. Since this is performed within `\@gls@link` the label can be accessed with `\glslabel`.

```
4879 \newcommand{\@glsxtr@do@inc@linkcount}{%
```

Does this entry have the linkcount attribute set?

```
4880 \glsifattribute{\glslabel}{linkcount}{true}%
```

```
4881 {%
```

Does the counter exist?

```
4882 \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
```

```
4883 {%
```

Counter doesn’t exist, so define it.

```
4884 \newcounter{glsxtr@linkcount@\glslabel}%
```

If linkcountmaster is set, add to counter reset.

```
4885 \glshasattribute{\glslabel}{linkcountmaster}%
```

```
4886 {%
```

Need to ensure values are fully expanded.

```
4887 \begingroup
```

```
4888 \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@\glslabel}%
```

```
4889 {\glsgetattribute{\glslabel}{linkcountmaster}}}%
```

```
4890 \x
```

```
4891 }%
```

```
4892 {}%
```

```
4893 }%
```

Increment counter:

```
4894 \glsxtrinlinkcounter{glsxtr@linkcount@\glslabel}%
```

```
4895 }%
```

```
4896 {}%
```

```
4897 }
```

`\rinc@linkcounter` May be redefined to use `\refstepcounter` if required.

```
4898 \newcommand*{\glsxtrinlinkcounter}[1]{\stepcounter{#1}}
```

`\linkCounterValue` Expands to the associated link counter register or 0 if not defined.

```
4899 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
```

```
4900 \ifcsundef{c@glsxtr@linkcount@#1}{0}{\csname c@glsxtr@linkcount@#1\endcsname}%
```

```
4901 }
```

`rTheLinkCounter` Expands to the display value of the associated link counter or 0 if not defined.

```
4902 \newcommand*\GlsXtrTheLinkCounter}[1]{%
4903   \ifcsundef{theglsxtr@linkcount@#1}{0}%
4904   {\csname theglsextr@linkcount@#1\endcsname}%
4905 }
```

`fLinkCounterDef` Tests if the counter has been defined

```
4906 \newcommand*\GlsXtrIfLinkCounterDef}[3]{%
4907   \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
4908 }
```

`LinkCounterName` Expands to the associated link counter name. (No check for existence.)

```
4909 \newcommand*\GlsXtrLinkCounterName}[1]{glsextr@linkcount@#1}
```

`ableLinkCounting` `\GlsXtrEnableLinkCounting[<master counter>]{<categories>}`

Enable link counting for the given categories.

```
4910 \newcommand*\GlsXtrEnableLinkCounting}[2][{}]{%
4911   \let\glsextr@inc@linkcount\@glsextr@do@inc@linkcount
4912   \@for\@glsextr@label:=#2\do
4913   {%
4914     \glsssetcategoryattribute{\@glsextr@label}{linkcount}{true}%
4915     \ifstrempy{#1}{}%
4916     {%
4917       \ifcsundef{c@#1}%
4918       {\@nocounterr{#1}}%
4919       {\glsssetcategoryattribute{\@glsextr@label}{linkcountmaster}{#1}}%
4920     }%
4921   }%
4922 }
4923 \@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.

```
4924 \@ifpackageloaded{glossaries-accsupp}
4925 {
```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```
4926   \newcommand*\glsaccessname}[1]{%
```

```

4927 \glsnameaccessdisplay
4928 {%
4929 \glstryname{#1}%
4930 }%
4931 {#1}%
4932 }

```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

4933 \newcommand*{\Glsaccessname}[1]{%
4934 \glsnameaccessdisplay
4935 {%
4936 \glstryname{#1}%
4937 }%
4938 {#1}%
4939 }

```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```

4940 \newcommand*{\GLSaccessname}[1]{%
4941 \glsnameaccessdisplay
4942 {%
4943 \mfirstucMakeUppercase{\glstryname{#1}}%
4944 }%
4945 {#1}%
4946 }

```

`\glsaccesstext` Display the text value (no link and no check for existence).

```

4947 \newcommand*{\glsaccesstext}[1]{%
4948 \glstextaccessdisplay
4949 {%
4950 \glstrytext{#1}%
4951 }%
4952 {#1}%
4953 }

```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```

4954 \newcommand*{\Glsaccesstext}[1]{%
4955 \glstextaccessdisplay
4956 {%
4957 \glstrytext{#1}%
4958 }%
4959 {#1}%
4960 }

```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```

4961 \newcommand*{\GLSaccesstext}[1]{%
4962 \glstextaccessdisplay

```

```

4963     {%
4964         \mfirstucMakeUppercase{\glentrytext{#1}}}%
4965     }%
4966     {#1}%
4967 }

```

glsaccessplural Display the plural value (no link and no check for existence).

```

4968 \newcommand*{\glsaccessplural}[1]{%
4969     \glspluralaccessdisplay
4970     {%
4971         \glentryplural{#1}%
4972     }%
4973     {#1}%
4974 }

```

Glsaccessplural Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

4975 \newcommand*{\Glsaccessplural}[1]{%
4976     \glspluralaccessdisplay
4977     {%
4978         \glentryplural{#1}%
4979     }%
4980     {#1}%
4981 }

```

GLSaccessplural Display the plural value (no link and no check for existence) converted to upper case.

```

4982 \newcommand*{\GLSaccessplural}[1]{%
4983     \glspluralaccessdisplay
4984     {%
4985         \mfirstucMakeUppercase{\glentryplural{#1}}}%
4986     }%
4987     {#1}%
4988 }

```

\glsaccessfirst Display the first value (no link and no check for existence).

```

4989 \newcommand*{\glsaccessfirst}[1]{%
4990     \glsfirstaccessdisplay
4991     {%
4992         \glentryfirst{#1}%
4993     }%
4994     {#1}%
4995 }

```

\Glsaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

4996 \newcommand*{\Glsaccessfirst}[1]{%
4997     \glsfirstaccessdisplay
4998     {%

```

```

4999      \Glentryfirst{#1}%
5000  }%
5001  {#1}%
5002  }

```

`\GLSaccessfirst` Display the first value (no link and no check for existence) converted to upper case.

```

5003  \newcommand*{\GLSaccessfirst}[1]{%
5004    \glsfirstaccessdisplay
5005    {%
5006      \mfirstucMakeUppercase{\glentryfirst{#1}}%
5007    }%
5008    {#1}%
5009  }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```

5010  \newcommand*{\glsaccessfirstplural}[1]{%
5011    \glsfirstpluralaccessdisplay
5012    {%
5013      \glentryfirstplural{#1}%
5014    }%
5015    {#1}%
5016  }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

5017  \newcommand*{\Glsaccessfirstplural}[1]{%
5018    \glsfirstpluralaccessdisplay
5019    {%
5020      \Glentryfirstplural{#1}%
5021    }%
5022    {#1}%
5023  }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) converted to upper case.

```

5024  \newcommand*{\GLSaccessfirstplural}[1]{%
5025    \glsfirstpluralaccessdisplay
5026    {%
5027      \mfirstucMakeUppercase{\glentryfirstplural{#1}}%
5028    }%
5029    {#1}%
5030  }

```

`glsaccesssymbol` Display the symbol value (no link and no check for existence).

```

5031  \newcommand*{\glsaccesssymbol}[1]{%
5032    \glssymbolaccessdisplay
5033    {%
5034      \glentrysymbol{#1}%
5035    }%

```

```

5036     {#1}%
5037 }

```

GLsaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

5038 \newcommand*{\GLsaccesssymbol}[1]{%
5039     \glssymbolaccessdisplay
5040     {%
5041         \Glentrysymbol{#1}%
5042     }%
5043     {#1}%
5044 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

5045 \newcommand*{\GLSaccesssymbol}[1]{%
5046     \glssymbolaccessdisplay
5047     {%
5048         \mfirstucMakeUppercase{\Glentrysymbol{#1}}%
5049     }%
5050     {#1}%
5051 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```

5052 \newcommand*{\glssaccesssymbolplural}[1]{%
5053     \glssymbolpluralaccessdisplay
5054     {%
5055         \glentrysymbolplural{#1}%
5056     }%
5057     {#1}%
5058 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

5059 \newcommand*{\GLsaccesssymbolplural}[1]{%
5060     \glssymbolpluralaccessdisplay
5061     {%
5062         \Glentrysymbolplural{#1}%
5063     }%
5064     {#1}%
5065 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) converted to upper case.

```

5066 \newcommand*{\GLSaccesssymbolplural}[1]{%
5067     \glssymbolpluralaccessdisplay
5068     {%
5069         \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%
5070     }%
5071     {#1}%
5072 }

```

`\glsaccessdesc` Display the desc value (no link and no check for existence).

```

5073 \newcommand*{\glsaccessdesc}[1]{%
5074   \glsdescriptionaccessdisplay
5075   {%
5076     \glentrydesc{#1}%
5077   }%
5078   {#1}%
5079 }

```

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

5080 \newcommand*{\Glsaccessdesc}[1]{%
5081   \glsdescriptionaccessdisplay
5082   {%
5083     \Glsentrydesc{#1}%
5084   }%
5085   {#1}%
5086 }

```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```

5087 \newcommand*{\GLSaccessdesc}[1]{%
5088   \glsdescriptionaccessdisplay
5089   {%
5090     \mfirstucMakeUppercase{\glentrydesc{#1}}%
5091   }%
5092   {#1}%
5093 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence).

```

5094 \newcommand*{\glsaccessdescplural}[1]{%
5095   \glsdescriptionpluralaccessdisplay
5096   {%
5097     \glentrydescplural{#1}%
5098   }%
5099   {#1}%
5100 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

5101 \newcommand*{\Glsaccessdescplural}[1]{%
5102   \glsdescriptionpluralaccessdisplay
5103   {%
5104     \Glsentrydescplural{#1}%
5105   }%
5106   {#1}%
5107 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

5108 \newcommand*{\GLSaccessdescplural}[1]{%
5109   \glsdescriptionpluralaccessdisplay
5110   {%
5111     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5112   }%
5113   {#1}%
5114 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

5115 \newcommand*{\glsaccessshort}[1]{%
5116   \glsshortaccessdisplay
5117   {%
5118     \glsentryshort{#1}%
5119   }%
5120   {#1}%
5121 }

```

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5122 \newcommand*{\Glsaccessshort}[1]{%
5123   \glsshortaccessdisplay
5124   {%
5125     \Glsentryshort{#1}%
5126   }%
5127   {#1}%
5128 }

```

`\GLSaccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

5129 \newcommand*{\GLSaccessshort}[1]{%
5130   \glsshortaccessdisplay
5131   {%
5132     \mfirstucMakeUppercase{\glsentryshort{#1}}%
5133   }%
5134   {#1}%
5135 }

```

`\lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

5136 \newcommand*{\lsaccessshortpl}[1]{%
5137   \glsshortpluralaccessdisplay
5138   {%
5139     \glsentryshortpl{#1}%
5140   }%
5141   {#1}%
5142 }

```

`\Lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5143 \newcommand*{\Lsaccessshortpl}[1]{%

```

```

5144 \glsshortpluralaccessdisplay
5145 {%
5146 \Glsentryshortpl{#1}%
5147 }%
5148 {#1}%
5149 }

```

LSaccessshortpl Display the shortplural value (no link and no check for existence) converted to upper case.

```

5150 \newcommand*{\LSaccessshortpl}[1]{%
5151 \glsshortpluralaccessdisplay
5152 {%
5153 \mfirstucMakeUppercase{\Glsentryshortpl{#1}}%
5154 }%
5155 {#1}%
5156 }

```

\glsaccesslong Display the long form (no link and no check for existence).

```

5157 \newcommand*{\glsaccesslong}[1]{%
5158 \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5159 }

```

\Glsaccesslong Display the long form (no link and no check for existence).

```

5160
5161 \newcommand*{\Glsaccesslong}[1]{%
5162 \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5163 }

```

\GLSaccesslong Display the long value (no link and no check for existence) converted to upper case.

```

5164 \newcommand*{\GLSaccesslong}[1]{%
5165 \glslongaccessdisplay
5166 {%
5167 \mfirstucMakeUppercase{\Glsentrylong{#1}}%
5168 }%
5169 {#1}%
5170 }

```

glsaccesslongpl Display the long plural form (no link and no check for existence).

```

5171 \newcommand*{\glsaccesslongpl}[1]{%
5172 \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5173 }

```

Glsaccesslongpl Display the long plural form (no link and no check for existence).

```

5174
5175 \newcommand*{\Glsaccesslongpl}[1]{%
5176 \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5177 }

```

GLSaccesslongpl Display the longplural value (no link and no check for existence) converted to upper case.

```
5178 \newcommand*{\GLSaccesslongpl}[1]{%
5179 \glslongpluralaccessdisplay
5180 {%
5181 \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5182 }%
5183 {#1}%
5184 }
```

End of if part

```
5185 }
5186 {
```

No accessibility support. Just define these commands to do `\glsentry<xxx>`

\glsaccessname Display the name value (no link and no check for existence).

```
5187 \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.

```
5188 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}
```

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.

```
5189 \newcommand*{\GLSaccessname}[1]{%
5190 \protect\mfirstucMakeUppercase{\glsentryname{#1}}}
```

\glsaccesstext Display the text value (no link and no check for existence).

```
5191 \newcommand*{\glsaccesstext}[1]{\glsentrytext{#1}}
```

\Glsaccesstext Display the text value (no link and no check for existence) with the first letter converted to upper case.

```
5192 \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}
```

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.

```
5193 \newcommand*{\GLSaccesstext}[1]{%
5194 \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}
```

glsaccessplural Display the plural value (no link and no check for existence).

```
5195 \newcommand*{\glsaccessplural}[1]{\glsentryplural{#1}}
```

Glsaccessplural Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```
5196 \newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}
```

GLSaccessplural Display the plural value (no link and no check for existence). converted to upper case.

```
5197 \newcommand*{\GLSaccessplural}[1]{%
5198 \protect\mfirstucMakeUppercase{\glsentryplural{#1}}}
```

`\glsaccessfirst` Display the first value (no link and no check for existence).
5199 `\newcommand*{\glsaccessfirst}[1]{\glsentryfirst{#1}}`

`\Glsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.
5200 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.
5201 `\newcommand*{\GLSaccessfirst}[1]{%`
5202 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).
5203 `\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.
5204 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.
5205 `\newcommand*{\GLSaccessfirstplural}[1]{%`
5206 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`glsaccesssymbol` Display the symbol value (no link and no check for existence).
5207 `\newcommand*{\glsaccesssymbol}[1]{\glsentrysymbol{#1}}`

`Glsaccesssymbol` Display the symbol value (no link and no check for existence) with the first letter converted to upper case.
5208 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.
5209 `\newcommand*{\GLSaccesssymbol}[1]{%`
5210 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence).
5211 `\newcommand*{\glsaccesssymbolplural}[1]{\glsentrysymbolplural{#1}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.
5212 `\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence). converted to upper case.
5213 `\newcommand*{\GLSaccesssymbolplural}[1]{%`
5214 `\protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}`

`\glsaccessdesc` Display the desc value (no link and no check for existence).
5215 `\newcommand*{\glsaccessdesc}[1]{\glsentrydesc{#1}}`

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

5216 `\newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence). converted to upper case.

5217 `\newcommand*{\GLSaccessdesc}[1]{%`
5218 `\protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}`

`ccessdescplural` Display the descplural value (no link and no check for existence).

5219 `\newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}`

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

5220 `\newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}`

`ccessdescplural` Display the descplural value (no link and no check for existence). converted to upper case.

5221 `\newcommand*{\GLSaccessdescplural}[1]{%`
5222 `\protect\mfirstucMakeUppercase{\glsentrydescplural{#1}}}`

`\glsaccessshort` Display the short form (no link and no check for existence).

5223 `\newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}`

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

5224 `\newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}`

`\GLSaccessshort` Display the short value (no link and no check for existence). converted to upper case.

5225 `\newcommand*{\GLSaccessshort}[1]{%`
5226 `\protect\mfirstucMakeUppercase{\glsentryshort{#1}}}`

`lsaccessshortpl` Display the short plural form (no link and no check for existence).

5227 `\newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}`

`lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

5228 `\newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}`

`LSaccessshortpl` Display the shortplural value (no link and no check for existence). converted to upper case.

5229 `\newcommand*{\GLSaccessshortpl}[1]{%`
5230 `\protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}`

`\glsaccesslong` Display the long form (no link and no check for existence).

5231 `\newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}`

`\Glsaccesslong` Display the long form (no link and no check for existence).

5232 `\newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}`

`\GLSaccesslong` Display the long value (no link and no check for existence). converted to upper case.

```

5233 \newcommand*\GLSaccesslong}[1]{%
5234 \protect\mfirstucMakeUppercase{\glentrylong{#1}}}
```

`glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

5235 \newcommand*\glsaccesslongpl}[1]{\glentrylongpl{#1}}
```

`Glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

5236 \newcommand*\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

`GLSaccesslongpl` Display the longplural value (no link and no check for existence). converted to upper case.

```

5237 \newcommand*\GLSaccesslongpl}[1]{%
5238 \protect\mfirstucMakeUppercase{\glentrylongpl{#1}}}
```

End of else part

```

5239 }
```

1.6 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```

5240 \glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```

5241 \newcommand*\glsifcategory}[4]{%
5242 \ifglsfieldeq{#1}{category}{#2}{#3}{#4}%
5243 }
```

Categories can have attributes.

`categoryattribute` `\glssetcategoryattribute{<category>}{<attribute-label>}{<value>}`

Set (or override if already set) an attribute for the given category.

```

5244 \newcommand*\glssetcategoryattribute}[3]{%
5245 \csdef{@glxtr@categoryattr@#1@#2}{#3}%
5246 }
```

`categoryattribute` `\glsgetcategoryattribute{<category>}{<attribute-label>}`

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```

5247 \newcommand*\glsgetcategoryattribute}[2]{%
5248 \csuse{@glxtr@categoryattr@#1@#2}%
5249 }
```

categoryattribute `\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}`

Tests if the category has the given attribute set.

```
5250 \newcommand*{\glshascategoryattribute}[4]{%
5251   \ifcsvoid{@glstr@categoryattr@#1@#2}{#4}{#3}%
5252 }
```

\glissetattribute `\glissetattribute{<entry label>}{<attribute-label>}{<value>}`

Short cut where the category label is obtained from the entry information.

```
5253 \newcommand*{\glissetattribute}[3]{%
5254   \glsetcategoryattribute{\glscategory{#1}}{#2}{#3}%
5255 }
```

\glsggetattribute `\glsggetattribute{<entry label>}{<attribute-label>}`

Short cut where the category label is obtained from the entry information.

```
5256 \newcommand*{\glsggetattribute}[2]{%
5257   \glsgcategoryattribute{\glscategory{#1}}{#2}%
5258 }
```

\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.

```
5259 \newcommand*{\glshasattribute}[4]{%
5260   \ifglstryexists{#1}%
5261   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
5262   {#4}%
5263 }
```

categoryattribute `\glisifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

True if category has the attribute with the given value.

```
5264 \newcommand{\glisifcategoryattribute}[5]{%
```

```

5265 \ifcsundef{@glxtr@categoryattr@@#1@#2}%
5266 {#5}%
5267 {\ifcsstring{@glxtr@categoryattr@@#1@#2}{#3}{#4}{#5}}%
5268 }

```

`\glsifattribute` `\glsifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

Short cut to determine if the given entry has a category with the given attribute set.

```

5269 \newcommand{\glsifattribute}[5]{%
5270   \ifglentryexists{#1}%
5271   {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
5272   {#5}%
5273 }

```

Set attributes for the default general category:

```
5274 \glsetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
5275 \glsetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```

5276 \newcommand*{\glsetregularcategory}[1]{%
5277   \glsetcategoryattribute{#1}{regular}{true}%
5278 }

```

`ifregularcategory` `\glsifregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```

5279 \newcommand{\glsifregularcategory}[3]{%
5280   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
5281 }

```

`ifnotregularcategory` `\glsifnotregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```

5282 \newcommand{\glsifnotregularcategory}[3]{%
5283   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
5284 }

```

`\glsifregular` `\glsifregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to true.

```
5285 \newcommand{\glsifregular}[3]{%
5286   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
5287 }
```

`\glsifnotregular` `\glsifnotregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to false.

```
5288 \newcommand{\glsifnotregular}[3]{%
5289   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
5290 }
```

`\glsforeachincategory` `\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.

```
5291 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
5292   \forallglossaries[#1]{#3}%
5293   {%
5294     \forlentries[#3]{#4}%
5295     {%
5296       \glsifcategory{#4}{#2}{#5}{}%
5297     }%
5298   }%
5299 }
```

`\glsforeachwithattribute` `\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.

```

5300 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
5301   \forallglossaries[#1]{#4}%
5302   {%
5303     \forglsentries[#4]{#5}%
5304     {%
5305       \glsifattribute{#5}{#2}{#3}{#6}{}%
5306     }%
5307   }%
5308 }

```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to `index` and add `\glstrpostdescription`.

```

5309 \ifdef\newterm
5310 {%

```

`\newterm`

```

5311   \renewcommand*{\newterm}[2][ ]{%
5312     \newglossaryentry{#2}%
5313     {type={index},category=index,name={#2},%
5314      description={\glstrpostdescription\nopostdesc},#1}%
5315   }

```

Indexed terms are regular by default.

```

5316   \glsssetcategoryattribute{index}{regular}{true}

```

`trpostdescindex`

```

5317   \newcommand*{\glstrpostdescindex}{}

5318 }
5319 {}

```

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`.

```

5320 \ifdef\printsymbols
5321 {%

```

`glstrnewsymbol` Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```

5322   \newcommand*{\glstrnewsymbol}[3][ ]{%
5323     \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
5324   }

```

Symbols are regular by default.

```

5325   \glsssetcategoryattribute{symbol}{regular}{true}

```

`trpostdescsymbol`

```

5326   \newcommand*{\glstrpostdescsymbol}{}

```

```
5327 }
5328 {}
```

Similar for the numbers option.

```
5329 \ifdef\printnumbers
5330 {%
```

glsxtrnewnumber

```
5331 \ifdef\printnumbers
5332   \newcommand*\glsxtrnewnumber[3][]{%
5333     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
5334   }
```

Numbers are regular by default.

```
5335   \glssetcategoryattribute{number}{regular}{true}
```

rpostdescnumber

```
5336   \newcommand*\glsxtrpostdescnumber{}

5337 }
5338 {}
```

sxtrsetcategory Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
5339 \newcommand*\glsxtrsetcategory[2]{%
5340   \@for\@glsxtr@label:=#1\do
5341   {%
5342     \glsfieldxdef{\@glsxtr@label}{category}{#2}%
5343   }%
5344 }
```

tcategoryforall 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.

```
5345 \newcommand*\glsxtrsetcategoryforall[2]{%
5346   \forallglossaries[#1]{\@glsxtr@type}{%
5347     \forglentries[\@glsxtr@type]{\@glsxtr@label}%
5348     {%
5349       \glsfieldxdef{\@glsxtr@label}{category}{#2}%
5350     }%
5351   }%
5352 }
```

trfieldtitlecase

```
\glsxtrfieldtitlecase{\label}{\field}
```

Apply title casing to the contents of the given field.

```
5353 \newcommand*\glsxtrfieldtitlecase[2]{%
```

```

5354 \expandafter\glxtrfieldtitlecasecs\expandafter
5355   {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
5356 }

```

fieldtitlecasecs The command used by \glxtrfieldtitlecase. May be redefined to use a different command, for example, \xcapitalisefmtwords.

```

5357 \newcommand*{\glxtrfieldtitlecasecs}[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.

```

5358 \ifpackageloaded{glossaries-accsupp}
5359 {
5360   \renewcommand*{\glossentrydesc}[1]{%
5361     \glsdoifexistsorwarn{#1}%
5362     {%
5363       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```

5364   \glshasattribute{#1}{glossdescfont}%
5365   {%
5366     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
5367     \ifcsdef{\@glxtr@attrval}%
5368     {%
5369       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5370     }%
5371     {%
5372       \GlossariesExtraWarning{Unknown control sequence name
5373         ‘\@glxtr@attrval’ supplied in glossdescfont attribute
5374         for entry ‘#1’. Ignoring}%
5375       \let\@glxtr@glossdescfont\@firstofone
5376     }%
5377   }%
5378   {\let\@glxtr@glossdescfont\@firstofone}%
5379   \glsifattribute{#1}{glossdesc}{firstuc}%
5380   {%
5381     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
5382   }%
5383   {%
5384     \glsifattribute{#1}{glossdesc}{title}%
5385     {%
5386       \@glxtr@do@titlecaps@warn
5387       \glsdescriptionaccessdisplay
5388     {%
5389       \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
5390     }%
5391     {#1}%

```

```

5392     }%
5393     {%
5394         \@glxtr@glossdescfont{\glsaccessdesc{#1}}%
5395     }%
5396 }%
5397 }%
5398 }
5399 }
5400 {
5401 \renewcommand*{\glossentrydesc}[1]{%
5402     \glsdoifexistsorwarn{#1}%
5403     {%
5404         \glssetabbrvfmt{\glscategory{#1}}%
5405         \glsattribute{#1}{glossdescfont}%
5406         {%
5407             \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
5408             \ifcsdef{\@glxtr@attrval}%
5409             {%
5410                 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
5411             }%
5412             {%
5413                 \GlossariesExtraWarning{Unknown control sequence name
5414                     '\@glxtr@attrval' supplied in glossdescfont attribute
5415                     for entry '#1'. Ignoring}%
5416                 \let\@glxtr@glossdescfont\@firstofone
5417             }%
5418         }%
5419         {\let\@glxtr@glossdescfont\@firstofone}%
5420         \glsifattribute{#1}{glossdesc}{firstuc}%
5421         {%
5422             \@glxtr@glossdescfont{\Glsentrydesc{#1}}%
5423         }%
5424         {%
5425             \glsifattribute{#1}{glossdesc}{title}%
5426             {%
5427                 \@glxtr@do@titlecaps@warn
5428                 \@glxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
5429             }%
5430             {%
5431                 \@glxtr@glossdescfont{\glsentrydesc{#1}}%
5432             }%
5433         }%
5434     }%
5435 }
5436 }

```

`\glossentryname` If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

5437 `\@ifpackageloaded{glossaries-accsupp}`

```

5438 {
5439   \renewcommand*{\glossentryname}[1]{%
5440     \@glsdoifexistsorwarn{#1}%
5441     {%
5442       \glsetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

5443     \glshasattribute{#1}{glossnamefont}%
5444     {%
5445       \edef\@glsextr@attrval{\glsetattribute{#1}{glossnamefont}}%
5446       \ifcsdef{\@glsextr@attrval}%
5447       {%
5448         \letcs{\@glsextr@glossnamefont}{\@glsextr@attrval}%
5449       }%
5450     }%
5451     \GlossariesExtraWarning{Unknown control sequence name
5452       ‘\@glsextr@attrval’ supplied in glossnamefont attribute
5453       for entry ‘#1’. Reverting to default \string\glssnamefont}%
5454     \let\@glsextr@glossnamefont\glssnamefont
5455   }%
5456 }%
5457 {\let\@glsextr@glossnamefont\glssnamefont}%
5458 \glsifattribute{#1}{glossname}{firstuc}%
5459 {%
5460   \glsnameaccessdisplay
5461   {%
5462     \@glsextr@glossnamefont{\Glsentryname{#1}}%
5463   }%
5464   {#1}%
5465 }%
5466 {%
5467   \glsifattribute{#1}{glossname}{title}%
5468   {%
5469     \@glsextr@do@titlecaps@warn
5470     \glsnameaccessdisplay
5471     {%
5472       \@glsextr@glossnamefont{\glsextr@fieldtitlecase{#1}{name}}%
5473     }%
5474     {#1}%
5475   }%
5476   {%
5477     \glsifattribute{#1}{glossname}{uc}%
5478     {%
5479       \glsnameaccessdisplay
5480       {%

```

Hide the label from the upper-casing command.

```

5481       \letcs{\glo@name}{glo\@glsetoklabel{#1}@name}%
5482       \@glsextr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
5483     }%

```

```

5484         {#1}%
5485     }%
5486     {%
5487         \letcs{\glo@name}{\glo@\glsdetoklabel{#1}@name}%
5488         \glsnameaccessdisplay
5489         {%
5490             \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
5491             }%
5492         {#1}%
5493     }%
5494 }%
5495 }%

```

Do post-name hook:

```

5496     \glxtrpostnamehook{#1}%
5497 }%
5498 }
5499 }
5500 {
5501     \renewcommand*{\glossentryname}[1]{%
5502         \@glsdoifexistsorwarn{#1}%
5503         {%
5504             \glsetabbrvfmt{\glscategory{#1}}%
5505             \glshasattribute{#1}{glossnamefont}%
5506             {%
5507                 \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
5508                 \ifcsdef{\@glxtr@attrval}%
5509                 {%
5510                     \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5511                 }%
5512                 {%
5513                     \GlossariesExtraWarning{Unknown control sequence name
5514                     '\@glxtr@attrval' supplied in glossnamefont attribute
5515                     for entry '#1'. Reverting to default \string\glsnamefont}%
5516                     \let\@glxtr@glossnamefont\glsnamefont
5517                 }%
5518             }%
5519             {\let\@glxtr@glossnamefont\glsnamefont}%
5520             \gl@ifattribute{#1}{glossname}{firstuc}%
5521             {%
5522                 \@glxtr@glossnamefont{\Glsentryname{#1}}%
5523             }%
5524             {%
5525                 \gl@ifattribute{#1}{glossname}{title}%
5526                 {%
5527                     \@glxtr@do@titlecaps@warn
5528                     \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
5529                 }%
5530                 {%
5531                     \gl@ifattribute{#1}{glossname}{uc}%

```

```
5532      {%
```

Hide the label from the upper-casing command.

```
5533      \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
5534      \@glxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
5535      }%
5536      {%
```

This little trick is used by glossaries to allow the user to redefine `\glsnamefont` to use `\makefirstuc`. Support it even though they can now use the `firstuc` attribute.

```
5537      \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
5538      \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
5539      }%
5540      }%
5541      }%
```

Do post-name hook.

```
5542      \glxtrpostnamehook{#1}%
5543      }%
5544  }
5545 }
```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```
5546 \ifpackageloaded{glossaries-accsupp}
5547 {
5548   \renewcommand*{\Glossentryname}[1]{%
5549     \@glstdoifexistsorwarn{#1}%
5550     {%
5551       \glsetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the `glossnamefont` attribute to determine the font applied.

```
5552     \glshasattribute{#1}{glossnamefont}%
5553     {%
5554       \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
5555       \ifcsdef{\@glxtr@attrval}%
5556       {%
5557         \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5558         }%
5559         {%
5560           \GlossariesExtraWarning{Unknown control sequence name
5561             ‘\@glxtr@attrval’ supplied in glossnamefont attribute
5562             for entry ‘#1’. Reverting to default \string\glsnamefont}%
5563           \let\@glxtr@glossnamefont\glsnamefont
5564           }%
5565         }%
5566         {\let\@glxtr@glossnamefont\glsnamefont}%
5567         \glsnameaccessdisplay
5568         {%
5569           \@glxtr@glossnamefont{\Glsentryname{#1}}%
5570           }%
5571           {#1}%
```

Do post-name hook:

```
5572 \glxtrpostnamehook{#1}%
5573 }%
5574 }
5575 }
5576 {
5577 \renewcommand*{\Glossentryname}[1]{%
5578 \@glsdoifexistsorwarn{#1}%
5579 {%
5580 \glsetabbrvfmt{\glscategory{#1}}%
5581 \glshasattribute{#1}{glossnamefont}%
5582 {%
5583 \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
5584 \ifcsdef{\@glxtr@attrval}%
5585 {%
5586 \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5587 }%
5588 {%
5589 \GlossariesExtraWarning{Unknown control sequence name
5590 '\@glxtr@attrval' supplied in glossnamefont attribute
5591 for entry '#1'. Reverting to default \string\glnamefont}%
5592 \let\@glxtr@glossnamefont\glnamefont
5593 }%
5594 }%
5595 {\let\@glxtr@glossnamefont\glnamefont}%
5596 \@glxtr@glossnamefont{\Glsentryname{#1}}%
```

Do post-name hook:

```
5597 \glxtrpostnamehook{#1}%
5598 }%
5599 }
5600 }
```

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.

```
5601 \newcommand*{\glxtrpostnamehook}[1]{%
5602 \let\@glsnumberformat\@glxtr@defaultnumberformat
5603 \glxtrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
5604 \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
5605 \csuse{glxtrpostname\glscategory{#1}}%
5606 }
```

`trapostnamehook`

```
5607 \newcommand*{\glsextrapostnamehook}[1]{%
```

etaccessdisplay

```

5608 \ifpackageloaded{glossaries-accsupp}
5609 {
5610   \newcommand*{\glxtr@setaccessdisplay}[1]{%
5611     \ifcsdef{gls#1accessdisplay}%
5612       {\letcs\@glxtr@accessdisplay{gls#1accessdisplay}}%
5613       {%

```

This is essentially the reverse of `\@gl@fetchfield`, since the field supplied to `\glossentryname` has to be the internal label, but the `\gl<field>accessdisplay` commands use the key name.

```

5614     \edef\@gl@thisval{#1}%
5615     \@for\@gl@map:=\@gl@keymap\do{%
5616       \edef\@this@key{\expandafter\@secondoftwo\@gl@map}%
5617       \ifdefequal{\@this@key}{\@gl@thisval}%
5618       {%
5619         \edef\@gl@thisval{\expandafter\@firstoftwo\@gl@map}%
5620         \@endfortrue
5621       }%
5622     }%
5623   }%
5624   \ifcsdef{gls\@gl@thisval accessdisplay}%
5625     {\letcs\@glxtr@accessdisplay{gls\@gl@thisval accessdisplay}}%
5626     {\let\@glxtr@accessdisplay\@firstoftwo}%
5627   }%
5628 }
5629 }
5630 {%
5631   \newcommand*{\glxtr@setaccessdisplay}[1]{%
5632     \let\@glxtr@accessdisplay\@firstoftwo}
5633 }

```

`sentrynameother` Provide a command that works like `\glossentryname` but accesses a different field (which must be supplied using its internal field label).

```

5634 \newrobustcmd*{\glossentrynameother}[2]{%
5635   \@gl@doifexistsorwarn{#1}%
5636   {%

```

Accessibility support:

```

5637   \glxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

5638   \gl@setabbrvfmt{\gl@category{#1}}%
5639   \gl@hasattribute{#1}{glossnamefont}%
5640   {%
5641     \edef\@glxtr@attrval{\gl@getattribute{#1}{glossnamefont}}%
5642     \ifcsdef{\@glxtr@attrval}%
5643     {%
5644       \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
5645     }%

```

```

5646      {%
5647      \GlossariesExtraWarning{Unknown control sequence name
5648      '\@glsxtr@attrval' supplied in glossnamefont attribute
5649      for entry '#1'. Reverting to default \string\glsnamefont}%
5650      \let\@glsxtr@glossnamefont\glsnamefont
5651      }%
5652      }%
5653      {\let\@glsxtr@glossnamefont\glsnamefont}%
5654      \glsifattribute{#1}{glossname}{firstuc}%
5655      {%
5656      \@glsxtr@accessdisplay
5657      {\@glsxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
5658      {#1}%
5659      }%
5660      {%
5661      \glsifattribute{#1}{glossname}{title}%
5662      {%
5663      \@glsxtr@do@titlecaps@warn
5664      \@glsxtr@accessdisplay
5665      {\@glsxtr@glossnamefont{\@glsxtr@field@titlecase{#1}{#2}}}%
5666      {#1}%
5667      }%
5668      {%
5669      \glsifattribute{#1}{glossname}{uc}%
5670      {%
5671      \letcs{\glo@name}{glo@\@glsdetoklabel{#1}@#2}%
5672      \@glsxtr@accessdisplay
5673      {\@glsxtr@glossnamefont{\@mfirstucMakeUppercase{\glo@name}}}%
5674      {#1}%
5675      }%
5676      {%
5677      \letcs{\glo@name}{glo@\@glsdetoklabel{#1}@#2}%
5678      \@glsxtr@accessdisplay
5679      {\@expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}}%
5680      {#1}%
5681      }%
5682      }%
5683      }%

```

Do post-name hook.

```

5684      \glsxtrpostnamehook{#1}%
5685      }%
5686 }

```

`format@override` Determines if the format key should override the indexing attribute value.

```

5687 \newif\if@glsxtr@format@override
5688 \@glsxtr@format@overridefalse

```

If overriding is enabled, the `\glsnumber` command will have to be redefined in the index to use `\hyperpage` instead.

xFormatOverride

```
5689 \@ifpackageloaded{hyperref}
5690 {
    If hyperref's hyperindex option is on, then hyperref will automatically add \hyperpage, so
    don't add it.
5691   \ifHy@hyperindex
5692     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
5693       \@glstr@format@overridetrue
5694       \appto\theindex{\let\glshypernumber\@firstofone}%
5695     }
5696   \else
5697     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
5698       \@glstr@format@overridetrue
5699       \appto\theindex{\let\glshypernumber\hyperpage}%
5700     }
5701   \fi
5702 }
5703 {
5704   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
5705     \@glstr@format@overridetrue
5706   }
5707 }
5708 \@onlypreamble\GlsXtrEnableIndexFormatOverride
```

doautoindexname

```
5709 \newcommand*{\glstrdoautoindexname}[2]{%
5710   \glshasattribute{#1}{#2}%
5711   {%
    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.
5712     \@glstr@autoindex@setname{#1}%
    If the attribute value is simply "true" don't add an encap, otherwise use the value as the encap.
5713     \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
5714     \if@glstr@format@override
5715       \ifx\@glstr@numberformat\@glstr@defaultnumberformat
5716       \else
5717         \let\@glstr@attrval\@glstr@numberformat
5718       \fi
5719     \fi
5720     \ifdefstring{\@glstr@attrval}{true}%
5721     {}%
5722     {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
5723     \expandafter\glstrautoindex\expandafter{\@glo@name}%
5724   }%
5725   {}%
5726 }
```

glxstrautoindex

```
5727 \newcommand*{\glxstrautoindex}{\index}
```

toindex@setname Assign \@glo@name for use with indexname attribute.

```
5728 \newcommand*{\@glxstr@autoindex@setname}[1]{%
5729   \protected@edef\@glo@name{\glxstrautoindexentry{#1}}%
5730   \glxstrautoindexassignsort{\@glo@sort}{#1}%
5731   \@gls@checkmkidxchars\@glo@sort
5732   \@glxstr@autoindex@doextra@esc\@glo@sort
5733   \epreto\@glo@name{\@glo@sort\@glxstr@autoindex@at}%
5734 }
```

rautoindexentry Command used for the actual part when auto-indexing.

```
5735 \newcommand*{\glxstrautoindexentry}[1]{\string\glsentryname{#1}}
```

indexassignsort Used to assign the sort value when auto-indexing.

```
5736 \newcommand*{\glxstrautoindexassignsort}[2]{%
5737   \glsletentryfield{#1}{#2}{sort}%
5738 }
```

dex@doextra@esc

```
5739 \newcommand*{\@glxstr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
5740   \ifx\@glxstr@autoindex@esc\@gls@quotechar
5741   \else
5742     \def\@gls@checkedmkidx{}%
5743     \edef\@glxstr@checkspch{%
5744       \noexpand\@glxstr@autoindex@escquote\expandonce{#1}%
5745       \noexpand\@empty\@glxstr@autoindex@esc\noexpand\@nnil
5746       \@glxstr@autoindex@esc\noexpand\@empty\noexpand\@glxstr@endescspch}%
5747     \@glxstr@checkspch
5748     \let#1\@gls@checkedmkidx\relax
5749   \fi
```

Escape actual character unless it has already been escaped.

```
5750   \ifx\@glxstr@autoindex@at\@gls@actualchar
5751   \else
5752     \def\@gls@checkedmkidx{}%
5753     \edef\@glxstr@checkspch{%
5754       \noexpand\@glxstr@autoindex@escat\expandonce{#1}%
5755       \noexpand\@empty\@glxstr@autoindex@at\noexpand\@nnil
5756       \@glxstr@autoindex@at\noexpand\@empty\noexpand\@glxstr@endescspch}%
5757     \@glxstr@checkspch
5758     \let#1\@gls@checkedmkidx\relax
5759   \fi
```

Escape level character unless it has already been escaped.

```
5760   \ifx\@glxstr@autoindex@level\@gls@levelchar
5761   \else
```

```

5762 \def\@gls@checkedmkidx{%
5763 \edef\@@glsxtr@checkspch{%
5764 \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
5765 \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
5766 \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
5767 @@glsxtr@checkspch
5768 \let#1\@gls@checkedmkidx\relax
5769 \fi

```

Escape encap character unless it has already been escaped.

```

5770 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
5771 \else
5772 \def\@gls@checkedmkidx{%
5773 \edef\@@glsxtr@checkspch{%
5774 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
5775 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
5776 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
5777 @@glsxtr@checkspch
5778 \let#1\@gls@checkedmkidx\relax
5779 \fi
5780 }

```

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`.

```

5781 \newcommand*{\@glsxtr@autoindex@at}{}

```

`\trSetActualChar` Set the actual character.

```

5782 \newcommand*{\GlsXtrSetActualChar}[1]{%
5783 \gdef\@glsxtr@autoindex@at{#1}%
5784 \def\@glsxtr@autoindex@escat##1##2##3\@glsxtr@endescspch{%
5785 \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
5786 }%
5787 }
5788 \@onlypreamble\GlsXtrSetActualChar
5789 \makeatother
5790 \GlsXtrSetActualChar{@}
5791 \makeatletter

```

`\autoindex@encap` Encap character for use with `\index`.

```

5792 \newcommand*{\@glsxtr@autoindex@encap}{}

```

`\XtrSetEncapChar` Set the encap character.

```

5793 \newcommand*{\GlsXtrSetEncapChar}[1]{%
5794 \gdef\@glsxtr@autoindex@encap{#1}%
5795 \def\@glsxtr@autoindex@escencap##1##2##3\@glsxtr@endescspch{%
5796 \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
5797 }%

```

```

5798 }
5799 \GlsXtrSetEncapChar{||}
5800 \@onlypreamble\GlsXtrSetEncapChar

autoindex@level  Level character for use with \index.
5801 \newcommand*{\@glstr@autoindex@level}{%

XtrSetLevelChar  Set the encap character.
5802 \newcommand*{\GlsXtrSetLevelChar}[1]{%
5803   \gdef\@glstr@autoindex@level{#1}%
5804   \def\@glstr@autoindex@esclevel##1##2##3\@glstr@endescspch{%
5805     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@esclevel}{##1}{##2}{##3}%
5806   }%
5807 }
5808 \GlsXtrSetLevelChar{!}
5809 \@onlypreamble\GlsXtrSetLevelChar

r@autoindex@esc  Escape character for use with \index.
5810 \newcommand*{\@glstr@autoindex@esc}{"}

lsXtrSetEscChar  Set the escape character.
5811 \newcommand*{\GlsXtrSetEscChar}[1]{%
5812   \gdef\@glstr@autoindex@esc{#1}%
5813   \def\@glstr@autoindex@escquote##1##2##3\@glstr@endescspch{%
5814     \@glstr@autoindex@escspch{#1}{\@glstr@autoindex@escquote}{##1}{##2}{##3}%
5815   }%
5816 }
5817 \GlsXtrSetEscChar{"}
5818 \@onlypreamble\GlsXtrSetEscChar

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:
5819 \ifdef\actualchar
5820 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
5821 {}

Quote character \quotechar:
5822 \ifdef\quotechar
5823 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
5824 {}

Level character \levelchar:
5825 \ifdef\levelchar
5826 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
5827 {}

Encap character \encapchar:
5828 \ifdef\encapchar
5829 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
5830 {}

```

leto@endescspch

```
5831 \def\@glxtr@gobbleto@endescspch#1\@glxtr@endescspch{}
```

toindex@esc@spch

```
\@glxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}
```

```
5832 \newcommand*{\@glxtr@autoindex@escspch}[5]{%
5833   \@glstmpb=\expandafter{\@glstmpb\the\@glstmpb}%
5834   \toks@={#3}%
5835   \ifx\@nnil#3\relax
5836     \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch#5\@glxtr@endescspch}%
5837   \else
5838     \ifx\@nnil#4\relax
5839       \edef\@glstmpb{\the\@glstmpb\the\@glstmpb}%
5840       \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch
5841         #4#5\@glxtr@endescspch}%
5842     \else
5843       \edef\@glstmpb{\the\@glstmpb\the\@glstmpb}%
5844       \def\@glxtr@autoindex@esc#1{%
5845         \def\@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
5846       }
5847     \fi
5848   \def\@glxtr@checkspch
5849 }
```

\Glossentrydesc Redefine to set the abbreviation format and accessibility support.

```
5850 \renewcommand*{\Glossentrydesc}[1]{%
5851   \glsoifexistsorwarn{#1}%
5852   {%
5853     \glsssetabbrvfmt{\glscategory{#1}}%
5854     \Glsaccessdesc{#1}%
5855   }%
5856 }
```

\lossentrysymbol Redefine to set the abbreviation format and accessibility support.

```
5857 \renewcommand*{\lossentrysymbol}[1]{%
5858   \glsoifexistsorwarn{#1}%
5859   {%
5860     \glsssetabbrvfmt{\glscategory{#1}}%
5861     \Glsaccesssymbol{#1}%
5862   }%
5863 }
```

\lossentrysymbol Redefine to set the abbreviation format and accessibility support.

```
5864 \renewcommand*{\lossentrysymbol}[1]{%
5865   \glsoifexistsorwarn{#1}%
5866   {%
```

```

5867 \glssetabbrvfmt{\glscategory{#1}}%
5868 \Glsaccesssymbol{#1}%
5869 }%
5870 }

```

Allow initials to be marked but only use the formatting for the tag in the glossary.

`\enableInitialTagging` 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.

```

5871 \newcommand*{\GlsXtrEnableInitialTagging}{%
5872 \ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
5873 }
5874 \@onlypreamble\GlsXtrEnableInitialTagging

```

`\r@enabletagging` Starred version undefines command.

```

5875 \newcommand*{\s@glxtr@enabletagging}[2]{%
5876 \undef#2%
5877 \@glxtr@enabletagging{#1}{#2}%
5878 }

```

`\r@enabletagging` Internal command.

```

5879 \newcommand*{\@glxtr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
5880 \@for\@glxtr@cat:=#1\do
5881 {%
5882 \ifdefempty\@glxtr@cat
5883 }%
5884 {\glssetcategoryattribute{\@glxtr@cat}{tagging}{true}}%
5885 }%
5886 \newrobustcmd*#2[1]{##1}%
5887 \def\@glxtr@taggingcs{#2}%
5888 \renewcommand*\@glxtr@activate@initialtagging{%
5889 \let#2\@glxtr@tag
5890 }%
5891 \ifundef\@gls@preglossaryhook
5892 {\GlossariesExtraWarning{Initial tagging requires at least
5893 glossaries.sty v4.19 to work correctly}}%
5894 }%
5895 }

```

Are we using an old version of `mfirstuc` that has a bug in `\capitalisewords`? If so, patch it so we don't have a problem with a combination of tagging and title case.

`\mfu@checkword@do` If this command hasn't been defined, then we have pre v2.02 of `mfirstuc`

```

5896 \ifundef\mfu@checkword@do
5897 {
5898 \newcommand*{\mfu@checkword@do}[1]{%

```

```

5899 \ifdefstring{\mfu@checkword@arg}{#1}%
5900 {%
5901     \let\@mfu@domakefirstuc\@firstofone
5902     \listbreak
5903 }%
5904 {}%
5905 }

```

`\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.

```

5906 \ifundef\mfu@checkword
5907 {
5908     \newcommand{\@glxtr@do@titlecaps@warn}{%
5909         \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
5910             support not available}%
5911         One warning should suffice.
5912     }
5913 }
5914 {
5915     \renewcommand*{\mfu@checkword}[1]{%
5916         \def\mfu@checkword@arg{#1}%
5917         \let\@mfu@domakefirstuc\makefirstuc
5918         \forlistloop\mfu@checkword@do\@mfu@nocaplist
5919     }
5920 }
5921 }
5922 {}% no patch required

```

`@titlecaps@warn` Do warning if title case not supported.

```

5923 \newcommand*{\@glxtr@do@titlecaps@warn}{}

```

`@initialtagging` Used in `\printglossary` but at least v4.19 of `glossaries` required.

```

5924 \newcommand*{\@glxtr@activate@initialtagging}{}

```

`\@glxtr@tag` Definition of tagging command when used in glossary.

```

5925 \newrobustcmd*{\@glxtr@tag}[1]{%
5926     \glxtr@tagfont{#1}{#1}%
5927     \glxtr@tagfont{#1}{#1}%
5928 }

```

`\glxtrtagfont` Used in the glossary.

```

5929 \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.

```

5930 \ifdef \@gls@preglossaryhook
5931 {
5932   \renewcommand*{\@gls@preglossaryhook}{%
5933     \@glsxtr@activate@initialtagging
      Since the glossaries are automatically scoped, \@glsxtr@org@postdescription shouldn't
      already be defined, but check anyway just as a precautionary measure.
5934     \ifundef \@glsxtr@org@postdescription
5935     {%
5936       \let \@glsxtr@org@postdescription \glspostdescription
5937       \renewcommand*{\glspostdescription}{%
5938         \ifglsentryexists{\glscurrententrylabel}%
5939         {%
5940           \glsxtrpostdescription
5941           \@glsxtr@org@postdescription
5942         }%
5943       }%
5944     }%
5945   }%
5946 }%

      Enable the options used by \@glsxtrp:
5947   \glossxtrsetpopts
5948 }%
5949 }
5950 {}

```

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.

```

5951 \newcommand*{\glsxtrpostdescription}{%
5952   \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
5953 }

```

postdescgeneral

```

5954 \newcommand*{\glsxtrpostdescgeneral}{}

```

xtrpostdescterm

```

5955 \newcommand*{\glsxtrpostdescterm}{}

```

postdescacronym

```

5956 \newcommand*{\glsxtrpostdescacronym}{}

```

escabbreviation

```

5957 \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.

```
5958 \renewcommand*{\glspostlinkhook}{%
5959   \ifglstryexists{\glslabel}{\glstrpostlinkhook}{}%
5960 }
```

`\glstrpostlinkhook` The entry label should already be stored in `\glslabel` by `\@gls@link`.

```
5961 \newcommand*{\glstrpostlinkhook}{%
5962   \glstrdiscardperiod{\glslabel}%
5963   {\glstrpostlinkendsentence}%
5964   {\glstrifcustomdiscardperiod
5965     {\glstrifperiod{\glstrpostlinkendsentence}{\glstrpostlink}}}%
5966   {\glstrpostlink}%
5967 }%
5968 }
```

`\glstrdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
5969 \newcommand*{\glstrifcustomdiscardperiod}[2]{#2}
```

`\glstrpostlink`

```
5970 \newcommand*{\glstrpostlink}{%
5971   \csuse{glstrpostlink\glscategory{\glslabel}}%
5972 }
```

`\glstrpostlinkendsentence` Done by `\glstrpostlinkhook` if a full stop is discarded.

```
5973 \newcommand*{\glstrpostlinkendsentence}{%
5974   \ifcsdef{glstrpostlink\glscategory{\glslabel}}
5975   {%
5976     \csuse{glstrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
5977   .\spacefactor\sfcode'\. \relax
5978 }%
5979 {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the spacefactor.

```
5980   \spacefactor\sfcode'\. \relax
5981 }%
5982 }
```

`\glstraddDescOnFirstUse` 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.

```
5983 \newcommand*{\glstrpostlinkAddDescOnFirstUse}{%
5984   \glstrifwasfirstuse{\space\glstrparen{\glssuccessdesc{\glslabel}}}{}%
5985 }
```

symbolOnFirstUse Provide a command for appending the symbol (if defined) in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```

5986 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
5987   \glxtrifwasfirstuse
5988   {%
5989     \ifglshassymbol{\glslabel}%
5990     {\space\glxtrparen{\glssaccesssymbol{\glslabel}}}%
5991     {}}%
5992   }%
5993   {}%
5994 }

```

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).

```

5995 \newcommand*{\glxtrdiscardperiod}[3]{%
5996   \glxtrifwasfirstuse
5997   {%
5998     \glusifattribute{#1}{retainfirstuseperiod}{true}%
5999     {#3}%
6000     {%
6001       \glusifattribute{#1}{discardperiod}{true}%
6002       {%
6003         \glusifplural
6004         {%
6005           \glusifattribute{#1}{pluraldiscardperiod}{true}%
6006           {\glxtrifperiod{#2}{#3}}%
6007           {#3}%
6008         }%
6009         {%
6010           \glxtrifperiod{#2}{#3}%
6011           }%
6012         }%
6013       {#3}%
6014     }%
6015   }%
6016   {%
6017     \glusifattribute{#1}{discardperiod}{true}%
6018     {%
6019       \glusifplural
6020       {%
6021         \glusifattribute{#1}{pluraldiscardperiod}{true}%
6022         {\glxtrifperiod{#2}{#3}}%
6023         {#3}%
6024       }%
6025       {%
6026         \glxtrifperiod{#2}{#3}%
6027         }%

```

```

6028 }%
6029 {#3}%
6030 }%
6031 }

```

`\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`

```

6032 \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 ' ').

```

6033 \newcommand*\glxtr@punclist{.,,:;?!}

```

`\punctuationmark` Add character to punctuation list.

```

6034 \newcommand*\glxtraddpunctuationmark}[1]{\appto\glxtr@punclist{#1}}

```

`\punctuationmarks` Reset the punctuation list.

```

6035 \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`.)

```

6036 \newcommand*\glxtrifnextpunc}[2]{%
6037   \def\reserved@a{#1}%
6038   \def\reserved@b{#2}%
6039   \futurelet\@glspunc@token\glxtr@ifnextpunc
6040 }

```

`\glxtr@ifnextpunc`

```

6041 \newcommand*\glxtr@ifnextpunc{%-
6042   \glxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{-}%
6043   \reserved@b
6044 }

```

`\glxtr@ifpunctoken` Test if the token given in the first argument is in the punctuation list.

```

6045 \newcommand*\glxtr@ifpunctoken}[1]{%
6046   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
6047 }

```

`\glxtr@ifpunctoken`

```

6048 \def\@glxtr@ifpunctoken#1#2{%
6049   \let\reserved@d=#2%
6050   \ifx\reserved@d\@nnil

```

```

6051 \let\glxstr@next\@glxstr@notfoundinlist
6052 \else
6053 \ifx#1\reserved@d
6054 \let\glxstr@next\@glxstr@foundinlist
6055 \else
6056 \let\glxstr@next\@glxstr@ifpunctoken
6057 \fi
6058 \fi
6059 \glxstr@next#1%
6060 }

```

tr@foundinlist

```

6061 \def\@glxstr@foundinlist#1\@nnil{\@firstoftwo}

```

@notfoundinlist

```

6062 \def\@glxstr@notfoundinlist#1{\@secondoftwo}

```

glxstrdopostpunc

```
\glxstrdopostpunc{<code>}
```

If this is followed by a punctuation character, do *<code>* after the character otherwise do *<code>* before whatever comes next.

```

6063 \newcommand{\glxstrdopostpunc}[1]{%
6064 \glxstr@ifnextpunc{\@glxstr@swaptwo{#1}}{#1}%
6065 }

```

@glxstr@swaptwo

```

6066 \newcommand{\@glxstr@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.

```

6067 \define@key{glxstrabbrv}{category}{%
6068 \edef\glscategorylabel{#1}%
6069 \ifcsdef{@glxabbrv@current@#1}%
6070 {%

```

Warning should already have been issued.

```

6071 \let\@glxstr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
6072 \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
6073 \glxstr@applyabbrvstyle{\csname @glxabbrv@current@#1\endcsname}%
6074 \let\GlsXtrWarnDeprecatedAbbrStyle\@glxstr@orgwarndep

```

```

6075 }%
6076 {}%
6077 }

```

Save the short plural form. This may be needed before the entry is defined.

```

6078 \define@key{glstrabbrv}{shortplural}{%
6079   \def\@gls@shortpl{#1}%
6080 }

```

Similarly for the long plural form.

```

6081 \define@key{glstrabbrv}{longplural}{%
6082   \def\@gls@longpl{#1}%
6083 }

```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

```

\glsshortpltok
6084 \newtoks\glsshortpltok

\glslongpltok
6085 \newtoks\glslongpltok

```

`\glstr@insertdots` Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to `\newabbreviation`. Note that explicitly using the short or shortplural keys will override this.

```

6086 \newcommand*{\@glstr@insertdots}[2]{%
6087   \def#1{}%
6088   \@glstr@insert@dots#1#2\@nnil
6089 }

```

```

\glstr@insert@dots
6090 \newcommand*{\@glstr@insert@dots}[2]{%
6091   \ifx\@nnil#2\relax
6092   \let\@glstr@insert@dots@next\@gobble
6093   \else
6094     \ifx\relax#2\relax
6095     \else
6096       \appto#1{#2.}%
6097     \fi
6098     \let\@glstr@insert@dots@next\@glstr@insert@dots
6099     \fi
6100     \@glstr@insert@dots@next#1%
6101 }

```

Similarly provide a way of replacing spaces with `\glstrwordsep`, which first needs to be defined:

```

\glxtrwordsep
6102 \newcommand*{\glxtrwordsep}{\space}

Each word is marked with

\glxtrword
6103 \newcommand*{\glxtrword}[1]{#1}

tr@markwordseps
6104 \newcommand*{\@glxtr@markwordseps}[2]{%
6105   \def#1{%
6106     \@glxtr@mark@wordseps#1#2 \@nnil
6107   }

r@mark@wordseps
6108 \def\@glxtr@mark@wordseps#1#2 #3{%
6109   \ifdefempty{#1}%
6110   {\def#1{\protect\glxtrword{#2}}}%
6111   {\appto#1{\protect\glxtrwordsep\protect\glxtrword{#2}}}%
6112   \ifx\@nnil#3\relax
6113   \let\@glxtr@mark@wordseps@next\relax
6114   \else
6115   \def\@glxtr@mark@wordseps@next{%
6116     \@glxtr@mark@wordseps#1#3}%
6117   \fi
6118   \@glxtr@mark@wordseps@next
6119 }

newabbreviation Define a new generic abbreviation.
6120 \newcommand*{\newabbreviation}[4][]{%
6121   \glxtr@newabbreviation{#1}{#2}{#3}{#4}%
6122 }

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.)
6123 \newcommand*{\glxtr@newabbreviation}[4]{%
6124   \glskeylisttok{#1}%
6125   \glslabeltok{#2}%
6126   \glsshorttok{#3}%
6127   \glslongtok{#4}%

Save the original short and long values (before attribute settings modify them).
6128   \def\glxtrorgshort{#3}%
6129   \def\glxtrorglong{#4}%

Get the category.
6130   \def\glscategorylabel{abbreviation}%
6131   \glxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%

```

Ignore the shortplural and longplural keys.

```
6132 \setkeys*{glsxtrabbrv}[shortplural,longplural]{#1}%
```

Set the default long plural

```
6133 \def\@gls@longpl{#4\glspluralsuffix}%
```

```
6134 \let\@gls@default@longpl\@gls@longpl
```

Has the markwords attribute been set?

```
6135 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
```

```
6136 {%
```

```
6137 \@glsxtr@markwordseps\@gls@long{#4}%
```

```
6138 \expandafter\def\expandafter\@gls@longpl\expandafter
```

```
6139 {\@gls@long\glspluralsuffix}%
```

```
6140 \let\@gls@default@longpl\@gls@longpl
```

Update \glslongtok.

```
6141 \expandafter\glslongtok\expandafter{\@gls@long}%
```

```
6142 }%
```

```
6143 {}%
```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```
6144 \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
```

```
6145 {%
```

```
6146 \@glsxtr@markwordseps\@gls@short{#3}%
```

```
6147 }%
```

```
6148 {}%
```

Has the insertdots attribute been set?

```
6149 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
```

```
6150 {%
```

```
6151 \@glsxtr@insertdots\@gls@short{#3}%
```

```
6152 \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
```

```
6153 }%
```

```
6154 {\def\@gls@short{#3}}%
```

```
6155 }%
```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```
6156 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
```

```
6157 {%
```

```
6158 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
```

```
6159 '\abbrvpluralsuffix}%
```

```
6160 }%
```

```
6161 {}%
```

Has the noshortplural attribute been set?

```
6162 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
```

```
6163 {%
```

```
6164 \let\@gls@shortpl\@gls@short
```

```
6165 }%
```

```
6166 {%
```

```
6167 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
```

```
6168 \abbrvpluralsuffix}%
```

```

6169     }%
6170 }%
    Update \glsshorttok:
6171 \expandafter\glsshorttok\expandafter{\@gls@short}%
    Hook for further customisation if required:
6172 \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).
6173 \setkeys*{glstrabbrv}[category]{#1}%
    Has the plural been explicitly set?
6174 \ifx\@gls@default@longpl\@gls@longpl
6175 \else
    Has the markwords attribute been set?
6176 \glscategoryattribute{\glscategorylabel}{markwords}{true}%
6177 {%
6178 \expandafter\@glstr@markwordseps\expandafter\@gls@longpl\expandafter
6179 {\@gls@longpl}%
6180 }%
6181 {}%
6182 \fi
    Set the plural token registers so the values can be accessed by the abbreviation styles.
6183 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
6184 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
    Do any extra setup provided by hook:
6185 \newabbreviationhook
    Define this entry:
6186 \protected@edef\@do@newglossaryentry{%
6187 \noexpand\newglossaryentry{\the\glslabeltok}%
6188 {%
6189 type=\glstrabbrvtype,%
6190 category=abbreviation,%
6191 short={\the\glsshorttok},%
6192 shortplural={\the\glsshortpltok},%
6193 long={\the\glslongtok},%
6194 longplural={\the\glslongpltok},%
6195 name={\the\glsshorttok},%
6196 \CustomAbbreviationFields,%
6197 \the\glskeylisttok
6198 }%
6199 }%
6200 \@do@newglossaryentry
6201 \GlsXtrPostNewAbbreviation
6202 }

```

evpresetkeyhook Hook for extra stuff in `\newabbreviation`
6203 `\newcommand*{\glxtrnewabbrevpresetkeyhook}[3]{}`

NewAbbreviation Hook used by abbreviation styles.
6204 `\newcommand*{\GlsXtrPostNewAbbreviation}{}`

bbreviationhook Hook for use with `\newabbreviation`.
6205 `\newcommand*{\newabbreviationhook}{}`

reviationFields
6206 `\newcommand*{\CustomAbbreviationFields}{}`

`\glxtrparen` For the parenthetical styles.
6207 `\newcommand*{\glxtrparen}[1]{(#1)}`

lsxtrfullformat Full format without case change.
6208 `\newcommand*{\glxtrfullformat}[2]{%`
6209 `\glsfirstlongfont{\glssaccesslong{#1}}#2\glxtrfullsep{#1}%`
6210 `\glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{#1}}}%`
6211 `}`

lsxtrfullformat Full format with case change.
6212 `\newcommand*{\Glsxtrfullformat}[2]{%`
6213 `\glsfirstlongfont{\Glsaccesslong{#1}}#2\glxtrfullsep{#1}%`
6214 `\glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshort{#1}}}%`
6215 `}`

xtrfullplformat Plural full format without case change.
6216 `\newcommand*{\glxtrfullplformat}[2]{%`
6217 `\glsfirstlongfont{\glssaccesslongpl{#1}}#2\glxtrfullsep{#1}%`
6218 `\glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshortpl{#1}}}%`
6219 `}`

xtrfullplformat Plural full format with case change.
6220 `\newcommand*{\Glsxtrfullplformat}[2]{%`
6221 `\glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glxtrfullsep{#1}%`
6222 `\glxtrparen{\protect\glsfirstabbrvfont{\glssaccessshortpl{#1}}}%`
6223 `}`

`\glxtrfullsep` Separator used by full format is a space by default. The argument is the entry's label.
6224 `\newcommand*{\glxtrfullsep}[1]{\space}`

In-line formats in case first use isn't compatible with `\gl Sentryfull` (for example, first use suppresses the long form or uses a footnote).

inlinefullformat Full format without case change.
6225 `\newcommand*{\glxtrinlinefullformat}{\glxtrfullformat}`

`\inlinefullformat` Full format with case change.
6226 `\newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}`

`\xtrfullplformat` Plural full format without case change.
6227 `\newcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}`

`\inefullplformat` Plural full format with case change.
6228 `\newcommand*{\Glsxtrinefullplformat}{\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`
6229 `\renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}`

`\Glsentryfull`
6230 `\renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}`

`\glsentryfullpl`
6231 `\renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}`

`\Glsentryfullpl`
6232 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinefullplformat{#1}{}}`

`\sfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.
6233 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\bbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.
6234 `\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.
6235 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\bbrvdefaultfont`
6236 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.
6237 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\longdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.
6238 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\lsfirstlongfont` Font changing command used for the long form on first use or in the full format.
6239 `\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}`

longdefaultfont

```
6240 \newcommand*{\glslongdefaultfont}[1]{\glslongdefaultfont{#1}}
```

brvpluralsuffix Default plural suffix. Allow an alternative default suffix for abbreviations.

```
6241 \newcommand*{\glxtrabbrvpluralsuffix}{\glspluralsuffix}
```

brvpluralsuffix Default plural suffix.

```
6242 \newcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}
```

\glxtrfull Full form (no case-change).

```
6243 \newrobustcmd*{\glxtrfull}{\@gls@hyp@opt\@ns@glxtrfull}
```

```
6244 \newcommand*\ns@glxtrfull[2][\%
```

```
6245 \new@ifnextchar[\@glxtr@full{#1}{#2}}%
```

```
6246 {\@glxtr@full{#1}{#2}[]}%
```

```
6247 }
```

\@glxtr@full Low-level macro:

```
6248 \def\@glxtr@full#1#2[#3]{%
```

```
6249 \glsdoifexists{#2}%
```

```
6250 {%
```

```
6251 \glsssetabbrvfmt{\glscategory{#2}}%
```

```
6252 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
```

```
6253 \let\gl@sifplural\@secondoftwo
```

```
6254 \let\gl@scapscase\@firstofthree
```

```
6255 \let\gl@insert\@empty
```

```
6256 \def\glscustomtext{\glxtrinlinefullformat{#2}{#3}}%
```

What should \glxtrifwasfirstuse be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

```
6257 \glxtrsetupfulldefs
```

```
6258 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
6259 }%
```

```
6260 \glspostlinkhook
```

```
6261 }
```

trsetupfulldefs

```
6262 \newcommand*{\glxtrsetupfulldefs}{%
```

```
6263 \let\glxtrifwasfirstuse\@firstoftwo
```

```
6264 }
```

\Glsxtrfull Full form (first letter uppercase).

```
6265 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\@ns@Glsxtrfull}
```

```
6266 \newcommand*\ns@Glsxtrfull[2][\%
```

```
6267 \new@ifnextchar[\@Glsxtr@full{#1}{#2}}%
```

```
6268 {\@Glsxtr@full{#1}{#2}[]}%
```

```
6269 }
```

`\@Glsxtr@full` Low-level macro:

```
6270 \def\@Glsxtr@full#1#2[#3]{%
6271   \glsdoifexists{#2}%
6272   {%
6273     \glssetabbrvfmt{\glscategory{#2}}%
6274     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6275     \let\glsifplural\@secondoftwo
6276     \let\glscapscase\@secondofthree
6277     \let\glsinsert\@empty
6278     \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
6279     \glsxtrsetupfulldefs
6280     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6281   }%
6282   \glspostlinkhook
6283 }
```

`\GLSxtrfull` Full form (all uppercase).

```
6284 \newrobustcmd*{\GLSxtrfull}{\@gl@hyp@opt\@ns@GLSxtrfull}
6285 \newcommand*\ns@GLSxtrfull[2][]{%
6286   \new@ifnextchar[{\@GLSxtr@full{#1}{#2}}%
6287   {\@GLSxtr@full{#1}{#2}[]}%
6288 }
```

`\@GLSxtr@full` Low-level macro:

```
6289 \def\@GLSxtr@full#1#2[#3]{%
6290   \glsdoifexists{#2}%
6291   {%
6292     \glssetabbrvfmt{\glscategory{#2}}%
6293     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6294     \let\glsifplural\@secondoftwo
6295     \let\glscapscase\@thirdofthree
6296     \let\glsinsert\@empty
6297     \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
6298     \glsxtrsetupfulldefs
6299     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6300   }%
6301   \glspostlinkhook
6302 }
```

`\glsxtrfullpl` Plural full form (no case-change).

```
6303 \newrobustcmd*{\glsxtrfullpl}{\@gl@hyp@opt\@ns@glsxtrfullpl}
6304 \newcommand*\ns@glsxtrfullpl[2][]{%
6305   \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}%
6306   {\@glsxtr@fullpl{#1}{#2}[]}%
6307 }
```

`\@glsxtr@fullpl` Low-level macro:

```
6308 \def\@glsxtr@fullpl#1#2[#3]{%
6309   \glsdoifexists{#2}%
6310 }
```

```

6310 {%
6311   \glssetabbrvfmt{\glscategory{#2}}%
6312   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6313   \let\glsifplural\@firstoftwo
6314   \let\gls caps case\@firstofthree
6315   \let\glsinsert\@empty
6316   \def\gls custom text{\glsxtr inline full pl format{#2}{#3}}%
6317   \glsxtr setup full defs
6318   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6319 }%
6320 \gls post link hook
6321 }

```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```

6322 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
6323 \newcommand*\ns@Glsxtrfullpl[2][{}]{%
6324   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
6325     {\@Glsxtr@fullpl{#1}{#2}[]}%
6326 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

6327 \def\@Glsxtr@fullpl#1#2[#3]{%
6328   \glsdoifexists{#2}%
6329   {%
6330     \glssetabbrvfmt{\glscategory{#2}}%
6331     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6332     \let\glsifplural\@firstoftwo
6333     \let\gls caps case\@secondofthree
6334     \let\glsinsert\@empty
6335     \def\gls custom text{\Glsxtr inline full pl format{#2}{#3}}%
6336     \glsxtr setup full defs
6337     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6338   }%
6339   \gls post link hook
6340 }

```

`\GLSxtrfullpl` Plural full form (all upper case).

```

6341 \newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\@ns@GLSxtrfullpl}
6342 \newcommand*\ns@GLSxtrfullpl[2][{}]{%
6343   \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}%
6344     {\@GLSxtr@fullpl{#1}{#2}[]}%
6345 }

```

`\@GLSxtr@fullpl` Low-level macro:

```

6346 \def\@GLSxtr@fullpl#1#2[#3]{%
6347   \glsdoifexists{#2}%
6348   {%
6349     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6350     \let\glsifplural\@firstoftwo

```

```

6351 \let\glscapscase\@thirdofthree
6352 \let\glsinsert\@empty
6353 \def\glscustomtext{%
6354     \mfirstucMakeUppercase{\glxtrinlinefullplformat{#2}{#3}}}%
6355 \glxtrsetupfulldefs
6356 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6357 }%
6358 \glspostlinkhook
6359 }

```

The short and long forms work in a similar way to acronyms.

`\glxtrshort`

```

6360 \newrobustcmd*{\glxtrshort}{\@gls@hyp@opt\@ns@glxtrshort}

```

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

```

6361 \newcommand*{\ns@glxtrshort}[2] [] {%
6362     \new@ifnextchar[{\@glxtrshort{#1}{#2}}{\@glxtrshort{#1}{#2} []}%
6363 }

```

Read in the final optional argument:

```

6364 \def\@glxtrshort#1#2[#3] {%
6365     \glsoifexists{#2}%
6366     {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

6367     \glssetabbrvfmt{\glscategory{#2}}%
6368     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6369     \let\glxtrifwasfirstuse\@secondoftwo
6370     \let\glsifplural\@secondoftwo
6371     \let\glscapscase\@firstofthree
6372     \let\glsinsert\@empty
6373     \def\glscustomtext{%
6374         \glsabbrvfont{\glsaccessshort{#2}\ifglxtrininsertinside#3\fi}%
6375         \ifglxtrininsertinside\else#3\fi
6376     }%
6377     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6378 }%
6379 \glspostlinkhook
6380 }

```

`\Glsxtrshort`

```

6381 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\@ns@Glsxtrshort}

```

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

```

6382 \newcommand*{\ns@Glsxtrshort}[2] [] {%
6383     \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
6384 }

```

Read in the final optional argument:

```

6385 \def\@Glsxtrshort#1#2[#3] {%

```

```

6386 \glsdoifexists{#2}%
6387 {%
6388   \glssetabbrvfmt{\glscategory{#2}}%
6389   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6390   \let\glstrifwasfirstuse\@secondoftwo
6391   \let\glsifplural\@secondoftwo
6392   \let\glscapscase\@secondofthree
6393   \let\glsinsert\@empty
6394   \def\glscustomtext{%
6395     \glsabbrvfont{\Glsaccessshort{#2}\ifglstrinsertinside#3\fi}%
6396     \ifglstrinsertinside\else#3\fi
6397   }%
6398   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6399 }%
6400 \glspostlinkhook
6401 }

```

\GLSxtrshort

```

6402 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\@ns@GLSxtrshort}

```

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

```

6403 \newcommand*{\ns@GLSxtrshort}[2][\@GLSxtrshort]{%
6404   \new@ifnextchar[\@GLSxtrshort{#1}{#2}]{\@GLSxtrshort{#1}{#2}[]}%
6405 }

```

Read in the final optional argument:

```

6406 \def\@GLSxtrshort#1#2[#3]{%
6407   \glsdoifexists{#2}%
6408   {%
6409     \glssetabbrvfmt{\glscategory{#2}}%
6410     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6411     \let\glstrifwasfirstuse\@secondoftwo
6412     \let\glsifplural\@secondoftwo
6413     \let\glscapscase\@thirdofthree
6414     \let\glsinsert\@empty
6415     \def\glscustomtext{%
6416       \mfirstucMakeUppercase
6417       {\glsabbrvfont{\Glsaccessshort{#2}\ifglstrinsertinside#3\fi}%
6418       \ifglstrinsertinside\else#3\fi
6419     }%
6420   }%
6421   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6422 }%
6423 \glspostlinkhook
6424 }

```

\glsxtrlong

```

6425 \newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\@ns@glsxtrlong}

```

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

```

6426 \newcommand*{\ns@glxtrlong}[2] [] {%
6427   \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
6428 }

```

Read in the final optional argument:

```

6429 \def\@glxtrlong#1#2[#3]{%
6430   \glsd@ifexists{#2}%
6431   {%
6432     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6433     \let\glxtrifwasfirstuse\@secondoftwo
6434     \let\gl@ifplural\@secondoftwo
6435     \let\glscapscase\@firstofthree
6436     \let\glinsert\@empty
6437     \def\glscustomtext{%
6438       \glslongfont{\gl@saccesslong{#2}\ifglxtrininsertinside#3\fi}%
6439       \ifglxtrininsertinside\else#3\fi
6440     }%
6441     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6442   }%
6443   \glspostlinkhook
6444 }

```

\Glsxtrlong

```

6445 \newrobustcmd*{\Glsxtrlong}{\@gl@hyp@opt\ns@Glsxtrlong}

```

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

```

6446 \newcommand*{\ns@Glsxtrlong}[2] [] {%
6447   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
6448 }

```

Read in the final optional argument:

```

6449 \def\@Glsxtrlong#1#2[#3]{%
6450   \glsd@ifexists{#2}%
6451   {%
6452     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6453     \let\glxtrifwasfirstuse\@secondoftwo
6454     \let\gl@ifplural\@secondoftwo
6455     \let\glscapscase\@secondofthree
6456     \let\glinsert\@empty
6457     \def\glscustomtext{%
6458       \glslongfont{\Glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
6459       \ifglxtrininsertinside\else#3\fi
6460     }%
6461     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6462   }%
6463   \glspostlinkhook
6464 }

```

\GLSxtrlong

```

6465 \newrobustcmd*{\GLSxtrlong}{\@gl@hyp@opt\ns@GLSxtrlong}

```

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

```
6466 \newcommand*{\ns@GLSxtrlong}[2][{}]{%
6467   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2}[]}%
6468 }
```

Read in the final optional argument:

```
6469 \def\@GLSxtrlong#1#2[#3]{%
6470   \glstoifexists{#2}%
6471   {%
6472     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6473     \let\glxtrifwasfirstuse\@secondoftwo
6474     \let\gl@sifplural\@secondoftwo
6475     \let\glscapscase\@thirdofthree
6476     \let\gl$insert\@empty
6477     \def\glscustomtext{%
6478       \mfirstucMakeUppercase
6479       {\glslongfont{\gl@saccesslong{#2}\ifglxtrininsertinside#3\fi}%
6480       \ifglxtrininsertinside\else#3\fi
6481     }%
6482   }%
6483   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6484   }%
6485   \glspostlinkhook
6486 }
```

Plural short forms:

\glxtrshortpl

```
6487 \newrobustcmd*{\glxtrshortpl}{\@gl@hyp@opt\ns@glxtrshortpl}
```

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

```
6488 \newcommand*{\ns@glxtrshortpl}[2][{}]{%
6489   \new@ifnextchar[{\@glxtrshortpl{#1}{#2}}{\@glxtrshortpl{#1}{#2}[]}%
6490 }
```

Read in the final optional argument:

```
6491 \def\@glxtrshortpl#1#2[#3]{%
6492   \glstoifexists{#2}%
6493   {%
6494     \glissetabbrfmt{\glscategory{#2}}%
6495     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
6496     \let\glxtrifwasfirstuse\@secondoftwo
6497     \let\gl@sifplural\@firstoftwo
6498     \let\glscapscase\@firstofthree
6499     \let\gl$insert\@empty
6500     \def\glscustomtext{%
6501       \glsabbrvfont{\gl@saccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
6502       \ifglxtrininsertinside\else#3\fi
6503     }%
6504     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6505   }%
```

```

6506 \glspostlinkhook
6507 }

```

`\Glsxtrshortpl`

```

6508 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\ns@Glsxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6509 \newcommand*{\ns@Glsxtrshortpl}[2] [] {%
6510   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
6511 }

    Read in the final optional argument:
6512 \def\@Glsxtrshortpl#1#2[#3] {%
6513   \glsdoifexists{#2}%
6514   {%
6515     \glssetabbrvfmt{\glscategory{#2}}%
6516     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6517     \let\glxtrifwasfirstuse\@secondoftwo
6518     \let\glsifplural\@firstoftwo
6519     \let\glscapscase\@secondofthree
6520     \let\glsinsert\@empty
6521     \def\glscustomtext{%
6522       \glsabbrvfont{\Glsaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
6523       \ifglxtrininsertinside\else#3\fi
6524     }%
6525     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6526   }%
6527   \glspostlinkhook
6528 }

```

`\GLSxtrshortpl`

```

6529 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
6530 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
6531   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
6532 }

    Read in the final optional argument:
6533 \def\@GLSxtrshortpl#1#2[#3] {%
6534   \glsdoifexists{#2}%
6535   {%
6536     \glssetabbrvfmt{\glscategory{#2}}%
6537     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6538     \let\glxtrifwasfirstuse\@secondoftwo
6539     \let\glsifplural\@firstoftwo
6540     \let\glscapscase\@thirdofthree
6541     \let\glsinsert\@empty
6542     \def\glscustomtext{%
6543       \mfirstucMakeUppercase
6544       {\glsabbrvfont{\Glsaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%

```

```

6545     \ifglxtrinsertinside\else#3\fi
6546   }%
6547 }%
6548   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6549 }%
6550 \glspostlinkhook
6551 }

```

Plural long forms:

\glxtrlongpl

```

6552 \newrobustcmd*{\glxtrlongpl}{\@gls@hyp@opt\ns@glxtrlongpl}

```

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

```

6553 \newcommand*{\ns@glxtrlongpl}[2] [] {%
6554   \new@ifnextchar[{\@glxtrlongpl{#1}{#2}}{\@glxtrlongpl{#1}{#2} []}%
6555 }

```

Read in the final optional argument:

```

6556 \def\@glxtrlongpl#1#2[#3] {%
6557   \glsdofexists{#2}%
6558   {%
6559     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6560     \let\glxtrifwasfirstuse\@secondoftwo
6561     \let\glsifplural\@firstoftwo
6562     \let\glscapscase\@firstofthree
6563     \let\glsinsert\@empty
6564     \def\glscustomtext{%
6565       \glslongfont{\glssaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
6566       \ifglxtrinsertinside\else#3\fi
6567     }%
6568     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6569   }%
6570   \glspostlinkhook
6571 }

```

\Glsxtrlongpl

```

6572 \newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\ns@Glsxtrlongpl}

```

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

```

6573 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
6574   \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
6575 }

```

Read in the final optional argument:

```

6576 \def\@Glsxtrlongpl#1#2[#3] {%
6577   \glsdofexists{#2}%
6578   {%
6579     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6580     \let\glxtrifwasfirstuse\@secondoftwo
6581     \let\glsifplural\@firstoftwo

```

```

6582 \let\glscapscase\@secondofthree
6583 \let\glsinsert\@empty
6584 \def\glscustomtext{%
6585 \glslongfont{\Glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
6586 \ifglxtrinsertinside\else#3\fi
6587 }%
6588 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6589 }%
6590 \glspostlinkhook
6591 }

```

\GLSxtrlongpl

```

6592 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}

Define the un-starred form. Need to determine if there is a final optional argument
6593 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
6594 \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
6595 }

Read in the final optional argument:
6596 \def\@GLSxtrlongpl#1#2[#3]{%
6597 \glsdoifexists{#2}%
6598 {%
6599 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
6600 \let\glxtrifwasfirstuse\@secondoftwo
6601 \let\glsifplural\@firstoftwo
6602 \let\glscapscase\@thirdofthree
6603 \let\glsinsert\@empty
6604 \def\glscustomtext{%
6605 \mfirstucMakeUppercase
6606 {\glslongfont{\Glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
6607 \ifglxtrinsertinside\else#3\fi
6608 }%
6609 }%
6610 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
6611 }%
6612 \glspostlinkhook
6613 }

```

\glssetabbrvfmt Set the current format for the given category (or the abbreviation category if unset).

```

6614 \newcommand*{\glssetabbrvfmt}[1]{%
6615 \ifcsdef{@glsabbrv@current@#1}%
6616 {\glxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
6617 {\glxtr@applyabbrvfmt{@glsabbrv@current@abbreviation}}%
6618 }

```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```

6619 \newrobustcmd*{\glsuseabbrvfont}[2]{\@glssetabbrvfmt{#2}\glsabbrvfont{#1}}

```

`\glsuselongfont` Provide a way to use the long font for a given category for arbitrary text.

```
6620 \newrobustcmd*{\glsuselongfont}[2]{\glssetabbrvfmt{#2}\glslongfont{#1}}
```

`sxtrgenabbrvfmt` Similar to `\glsngenacfmt`, but for abbreviations.

```
6621 \newcommand*{\glsxtrgenabbrvfmt}{%
6622   \ifdefempty\glscustomtext
6623   {%
6624     \ifglsused\glslabel
6625     {%
        Subsequent use:
6626       \glsifplural
6627       {%
        Subsequent plural form:
6628       \glscapscase
6629       {%
        Subsequent plural form, don't adjust case:
6630       \glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
6631       }%
6632       {%
        Subsequent plural form, make first letter upper case:
6633       \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
6634       }%
6635       {%
        Subsequent plural form, all caps:
6636       \mfirstucMakeUppercase
6637       {\glsxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
6638       }%
6639       }%
6640       {%
        Subsequent singular form
6641       \glscapscase
6642       {%
        Subsequent singular form, don't adjust case:
6643       \glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
6644       }%
6645       {%
        Subsequent singular form, make first letter upper case:
6646       \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
6647       }%
6648       {%
        Subsequent singular form, all caps:
6649       \mfirstucMakeUppercase
6650       {\glsxtrsubsequentfmt{\glslabel}{\glsinsert}}%
```

```

6651      }%
6652      }%
6653      }%
6654      {%

```

First use:

```

6655      \glsifplural
6656      {%

```

First use plural form:

```

6657      \glscapscase
6658      {%

```

First use plural form, don't adjust case:

```

6659      \glsxtrfullplformat{\glslabel}{\glsinsert}%
6660      }%
6661      {%

```

First use plural form, make first letter upper case:

```

6662      \Glsxtrfullplformat{\glslabel}{\glsinsert}%
6663      }%
6664      {%

```

First use plural form, all caps:

```

6665      \mfirstucMakeUppercase
6666      {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
6667      }%
6668      }%
6669      {%

```

First use singular form

```

6670      \glscapscase
6671      {%

```

First use singular form, don't adjust case:

```

6672      \glsxtrfullformat{\glslabel}{\glsinsert}%
6673      }%
6674      {%

```

First use singular form, make first letter upper case:

```

6675      \Glsxtrfullformat{\glslabel}{\glsinsert}%
6676      }%
6677      {%

```

First use singular form, all caps:

```

6678      \mfirstucMakeUppercase
6679      {\glsxtrfullformat{\glslabel}{\glsinsert}}%
6680      }%
6681      }%
6682      }%
6683      }%
6684      {%

```

User supplied text.

```
6685 \glscustomtext
6686 }%
6687 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
6688 \newcommand*{\glxtrsubsequentfmt}[2]{%
6689 \glabbrvfont{\glsaccessshort{#1}\ifglxtrinsertinside #2\fi}%
6690 \ifglxtrinsertinside \else#2\fi
6691 }
6692 \let\glxtrdefaultsubsequentfmt\glxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural no case change).

```
6693 \newcommand*{\glxtrsubsequentplfmt}[2]{%
6694 \glabbrvfont{\glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
6695 \ifglxtrinsertinside \else#2\fi
6696 }
6697 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt
```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).

```
6698 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
6699 \glabbrvfont{\Glsaccessshort{#1}\ifglxtrinsertinside #2\fi}%
6700 \ifglxtrinsertinside \else#2\fi
6701 }
6702 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural, first letter uppercase).

```
6703 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
6704 \glabbrvfont{\Glsaccessshortpl{#1}\ifglxtrinsertinside #2\fi}%
6705 \ifglxtrinsertinside \else#2\fi
6706 }
6707 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt
```

1.7.1 Abbreviation Styles Setup

breiviationstyle

```
6708 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
6709 \ifcsundef{@glabbrv@dispstyle@setup@#2}
6710 {%
6711 \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{%
6712 }%
6713 }%
```

Have abbreviations already been defined for this category?

```
6714 \ifcsstring{@glabbrv@current@#1}{#2}%
6715 {%
```

Style already set.

```
6716 }%
6717 {%
6718 \def\@glsxtr@dostylewarn{%
6719 \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
6720 {%
6721 \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
6722 style has been switched \MessageBreak
6723 for category ‘#1’, \MessageBreak
6724 but there have already been entries \MessageBreak
6725 defined for this category. Unwanted \MessageBreak
6726 side-effects may result}}%
6727 \@endfortrue
6728 }%
6729 \@glsxtr@dostylewarn
```

Set up the style for the given category.

```
6730 \csdef{@glsabbrv@current@#1}{#2}%
6731 \glsxtr@applyabbrvstyle{#2}%
6732 }%
6733 }%
6734 }
```

applyabbrvstyle Apply the abbreviation style without existence check.

```
6735 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
6736 \csuse{@glsabbrv@dispstyle@setup@#1}%
6737 \csuse{@glsabbrv@dispstyle@fmts@#1}%
6738 }
```

r@applyabbrvfmt Only apply the style formats.

```
6739 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
6740 \csuse{@glsabbrv@dispstyle@fmts@#1}%
6741 }
```

breivationstyle This is different from `\newacronymstyle`. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```
6742 \newcommand*{\newabbreviationstyle}[3]{%
6743 \ifcsdef{@glsabbrv@dispstyle@setup@#1}
6744 {%
6745 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
6746 defined}{}%
6747 }%
6748 {%
6749 \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```
6750 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
6751 #2}%
6752 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
6753 \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
6754 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
6755 \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
6756 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
```

Reset \glxtrsubsequentfmt etc in case a style changes this.

```
6757 \let\glxtrsubsequentfmt\glxtrdefaultsubsequentfmt
6758 \let\glxtrsubsequentplfmt\glxtrdefaultsubsequentplfmt
6759 \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
6760 \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
6761 #3}%
6762 }%
6763 }
```

breivationstyle

```
6764 \newcommand*{\renewabbreviationstyle}[3]{%
6765 \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
6766 {%
6767 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
6768 }%
6769 {%
6770 \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
6771 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
6772 #2}%
6773 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
6774 \renewcommand*{\glxtrinlinefullformat}{\glxtrfullformat}%
6775 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
6776 \renewcommand*{\glxtrinlinefullplformat}{\glxtrfullplformat}%
6777 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
6778 #3}%
6779 }%
6780 }
```

breivationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```
6781 \newcommand*{\letabbreviationstyle}[2]{%
6782 \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
6783 \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
6784 }
```

ecated@abbrstyle

```
\@glxtr@deprecated@abbrstyle{<old-name>}{<new-name>}
```

Define a synonym for a deprecated abbreviation style.

```
6785 \newcommand*{\@GlsXtrDeprecatedAbbrStyle}[2]{%
6786   \csdef{@Glsabbrv@dispstyle@setup@#1}{%
6787     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
6788     \csuse{@Glsabbrv@dispstyle@setup@#2}%
6789   }%
6790   \csletcs{@Glsabbrv@dispstyle@fmts@#1}{@Glsabbrv@dispstyle@fmts@#2}%
6791 }
```

`\GlsXtrWarnDeprecatedAbbrStyle` Generate warning for deprecated style use.

```
6792 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
6793   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
6794     use ‘#2’ instead}%
6795 }
```

`\GlsXtrUseAbbrStyleSetup`

```
6796 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
6797   \ifcsundef{@Glsabbrv@dispstyle@setup@#1}%
6798   {%
6799     \PackageError{glossaries-extra}%
6800     {Unknown abbreviation style definitions ‘#1’}{}%
6801   }%
6802   {%
6803     \csname @Glsabbrv@dispstyle@setup@#1\endcsname
6804   }%
6805 }
```

`\GlsXtrUseAbbrStyleFmts`

```
6806 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
6807   \ifcsundef{@Glsabbrv@dispstyle@fmts@#1}%
6808   {%
6809     \PackageError{glossaries-extra}%
6810     {Unknown abbreviation style formats ‘#1’}{}%
6811   }%
6812   {%
6813     \csname @Glsabbrv@dispstyle@fmts@#1\endcsname
6814   }%
6815 }
```

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.

```
6816 \newif\ifglxtrinsertinside
6817 \glxtrinsertinsidefalse
```

trlongshortname

```
6818 \newcommand*{\glxtrlongshortname}{%
6819   \protect\glxabbrvfont{\the\glsshorttok}%
6820 }
```

long-short

```
6821 \newabbreviationstyle{long-short}%
6822 {%
6823   \renewcommand*{\CustomAbbreviationFields}{%
6824     name={\glxtrlongshortname},
6825     sort={\the\glsshorttok},
6826     first={\protect\glxfirstlongfont{\the\glslongtok}%
6827       \protect\glxtrfullsep{\the\glslabeltok}%
6828       \glxtrparen{\protect\glxfirstabbrvfont{\the\glsshorttok}}},%
6829     firstplural={\protect\glxfirstlongfont{\the\glslongpltok}%
6830       \protect\glxtrfullsep{\the\glslabeltok}%
6831       \glxtrparen{\protect\glxfirstabbrvfont{\the\glsshortpltok}}},%
6832     plural={\protect\glxabbrvfont{\the\glsshortpltok}},%
6833     description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

```
6834 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6835   \glshasattribute{\the\glslabeltok}{regular}%
6836   {%
6837     \glissetattribute{\the\glslabeltok}{regular}{false}%
6838   }%
6839   {%
6840   }%
6841 }%
6842 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
6843 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
6844 \renewcommand*{\glxabbrvfont}[1]{\glxabbrvdefaultfont{##1}}%
6845 \renewcommand*{\glxfirstabbrvfont}[1]{\glxfirstabbrvdefaultfont{##1}}%
6846 \renewcommand*{\glxfirstlongfont}[1]{\glxfirstlongdefaultfont{##1}}%
6847 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
6848 \renewcommand*{\glxtrfullformat}[2]{%
6849   \glxfirstlongfont{\glxaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6850   \ifglxtrinsertinside\else##2\fi
6851   \glxtrfullsep{##1}}%
```

```

6852 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
6853 }%
6854 \renewcommand*{\glsxtrfullplformat}[2]{%
6855 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
6856 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
6857 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
6858 }%
6859 \renewcommand*{\Glsxtrfullformat}[2]{%
6860 \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
6861 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
6862 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
6863 }%
6864 \renewcommand*{\Glsxtrfullplformat}[2]{%
6865 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
6866 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
6867 \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
6868 }%
6869 }

```

Set this as the default style for general abbreviations:

```

6870 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

6871 \newcommand*{\glsxtrlongshortdescsort}{%
6872 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
6873 }

```

ngshortdescname

```

6874 \newcommand*{\glsxtrlongshortdescname}{%
6875 \protect\glslongfont{\the\glslongtok}
6876 \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}}%
6877 }

```

long-short-desc User supplies description. The long form is included in the name.

```

6878 \newabbreviationstyle{long-short-desc}%
6879 {%
6880 \renewcommand*{\CustomAbbreviationFields}{%
6881 name={\glsxtrlongshortdescname},
6882 sort={\glsxtrlongshortdescsort},%
6883 first={\protect\glsfirstlongfont{\the\glslongtok}%
6884 \protect\glsxtrfullsep{\the\glslabeltok}%
6885 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
6886 firstplural={\protect\glsfirstlongfont{\the\glslongtok}%
6887 \protect\glsxtrfullsep{\the\glslabeltok}%
6888 \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%

```

The text key should only have the short form.

```

6889 text={\protect\glsabbrvfont{\the\glsshorttok}},%

```

```

6890 plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
6891 }%

```

Unset the regular attribute if it has been set.

```

6892 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6893   \glshasattribute{\the\glslabeltok}{regular}%
6894   {%
6895     \glissetattribute{\the\glslabeltok}{regular}{false}%
6896   }%
6897   {}%
6898 }%
6899 }%
6900 {%
6901   \GlsXtrUseAbbrStyleFmts{long-short}%
6902 }

```

trshortlongname

```

6903 \newcommand*{\glxtrshortlongname}{%
6904   \protect\glsabbrvfont{\the\glsshorttok}%
6905 }

```

short-long Short form followed by long form in parenthesis on first use.

```

6906 \newabbreviationstyle{short-long}%
6907 {%
6908   \renewcommand*{\CustomAbbreviationFields}{%
6909     name={\glxtrshortlongname},
6910     sort={\the\glsshorttok},
6911     description={\the\glslongtok},%
6912     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
6913     \protect\glxtrfullsep{\the\glslabeltok}%
6914     \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
6915     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
6916     \protect\glxtrfullsep{\the\glslabeltok}%
6917     \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
6918     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

6919 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6920   \glshasattribute{\the\glslabeltok}{regular}%
6921   {%
6922     \glissetattribute{\the\glslabeltok}{regular}{false}%
6923   }%
6924   {}%
6925 }%
6926 }%
6927 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6928 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%

```

```

6929 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
6930 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
6931 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
6932 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

6933 \renewcommand*\glsxtrfullformat[2]{%
6934   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
6935   \ifglsxtrininsertinside\else##2\fi
6936   \glsxtrfullsep{##1}%
6937   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
6938 }%
6939 \renewcommand*\glsxtrfullplformat[2]{%
6940   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
6941   \ifglsxtrininsertinside\else##2\fi
6942   \glsxtrfullsep{##1}%
6943   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
6944 }%
6945 \renewcommand*\Glsxtrfullformat[2]{%
6946   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
6947   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
6948   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
6949 }%
6950 \renewcommand*\Glsxtrfullplformat[2]{%
6951   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
6952   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
6953   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
6954 }%
6955 }

```

ortlongdescsort

```

6956 \newcommand*\glsxtrshortlongdescsort{\the\glsshorttok}

```

ortlongdescname

```

6957 \newcommand*\glsxtrshortlongdescname{%
6958   \protect\glsabbrvfont{\the\glsshorttok}
6959   \glsxtrparen{\protect\glslongfont{\the\glslongtok}}%
6960 }

```

short-long-desc User supplies description. The long form is included in the name.

```

6961 \newabbreviationstyle{short-long-desc}%
6962 {%
6963   \renewcommand*\CustomAbbreviationFields{%
6964     name={\glsxtrshortlongdescname},
6965     sort={\glsxtrshortlongdescsort},
6966     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
6967       \protect\glsxtrfullsep{\the\glslabeltok}%
6968       \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
6969     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%

```

```

6970 \protect\glstrfullsep{\the\glslabeltok}%
6971 \glstrparen{\protect\glslfirstlongfont{\the\glslongpltok}}},%
6972 text={\protect\glssabrvfont{\the\glsshorttok}},%
6973 plural={\protect\glssabrvfont{\the\glsshortpltok}}}%
6974 }%

```

Unset the regular attribute if it has been set.

```

6975 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6976 \glshasattribute{\the\glslabeltok}{regular}%
6977 {%
6978 \glsssetattribute{\the\glslabeltok}{regular}{false}%
6979 }%
6980 {}%
6981 }%
6982 }%
6983 {%
6984 \GlsXtrUseAbbrStyleFmts{short-long}%
6985 }

```

`\glslongfootnotefont` Only used by the “footnote” styles.

```

6986 \newcommand*{\glslfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%

```

`\glslongfootnotefont` Only used by the “footnote” styles.

```

6987 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%

```

`\glsxtrabrvfootnote` `\glsxtrabrvfootnote{<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`).

```

6988 \newcommand*{\glsxtrabrvfootnote}[2]{\footnote{#2}}

```

`\glsxtrfootnotename`

```

6989 \newcommand*{\glsxtrfootnotename}{%
6990 \protect\glssabrvfont{\the\glsshorttok}%
6991 }

```

`\footnote` Short form followed by long form in footnote on first use.

```

6992 \newabbreviationstyle{footnote}%
6993 {%
6994 \renewcommand*{\CustomAbbreviationFields}{%
6995 name={\glsxtrfootnotename},
6996 sort={\the\glsshorttok},
6997 description={\the\glslongtok},%

```

```

6998 first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
6999 \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7000 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7001 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
7002 \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
7003 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
7004 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.

```

7005 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7006 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7007 \glsattribute{\the\glslabeltok}{regular}%
7008 {%
7009 \glssetattribute{\the\glslabeltok}{regular}{false}%
7010 }%
7011 {}%
7012 }%
7013 }%
7014 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7015 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
7016 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7017 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7018 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
7019 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7020 \renewcommand*{\glsxtrfullformat}[2]{%
7021 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7022 \ifglsxtrininsertinside\else##2\fi
7023 \protect\glsxtrabbrvfootnote{##1}%
7024 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7025 }%
7026 \renewcommand*{\glsxtrfullplformat}[2]{%
7027 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
7028 \ifglsxtrininsertinside\else##2\fi
7029 \protect\glsxtrabbrvfootnote{##1}%
7030 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7031 }%
7032 \renewcommand*{\Glsxtrfullformat}[2]{%
7033 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
7034 \ifglsxtrininsertinside\else##2\fi
7035 \protect\glsxtrabbrvfootnote{##1}%
7036 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
7037 }%
7038 \renewcommand*{\Glsxtrfullplformat}[2]{%
7039 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%

```

```

7040 \ifglxtrinsertinside\else##2\fi
7041 \protect\glxtrabbrvfootnote{##1}%
7042 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7043 }%

```

The first use full form and the inline full form use the short (long) style.

```

7044 \renewcommand*{\glxtrinlinefullformat}[2]{%
7045 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7046 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7047 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7048 }%
7049 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7050 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7051 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7052 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7053 }%
7054 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7055 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7056 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7057 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7058 }%
7059 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7060 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7061 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7062 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7063 }%
7064 }

```

short-footnote

```

7065 \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.

```

7066 \newabbreviationstyle{postfootnote}%
7067 {%
7068 \renewcommand*{\CustomAbbreviationFields}{%
7069 name={\glxtrfootnotename},
7070 sort={\the\glsshorttok},
7071 description={\the\glslongtok},%
7072 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
7073 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
7074 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.

```

7075 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7076 \csdef{glxtrpostlink\glscategorylabel}{%
7077 \glxtrifwasfirstuse

```

7078 {%

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
7079        \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7080        {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
7081        }%
7082        {%
7083        }%
7084        \glshasattribute{\the\glslabeltok}{regular}%
7085        {%
7086        \glissetattribute{\the\glslabeltok}{regular}{false}%
7087        }%
7088        {%
7089        }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
7090    \renewcommand*{\glxtrsetupfulldefs}{%
7091    \let\glxtrifwasfirstuse\@secondoftwo
7092    }%
7093 }%
7094 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
7095    \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7096    \renewcommand*{\glabbrvfont}[1]{\glabbrvdefaultfont{##1}}%
7097    \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7098    \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
7099    \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
7100    \renewcommand*{\glxtrfullformat}[2]{%
7101    \glsfirstabbrvfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7102    \ifglxtrininsertinside\else##2\fi
7103    }%
7104    \renewcommand*{\glxtrfullplformat}[2]{%
7105    \glsfirstabbrvfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7106    \ifglxtrininsertinside\else##2\fi
7107    }%
7108    \renewcommand*{\Glsxtrfullformat}[2]{%
7109    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7110    \ifglxtrininsertinside\else##2\fi
7111    }%
7112    \renewcommand*{\Glsxtrfullplformat}[2]{%
7113    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7114    \ifglxtrininsertinside\else##2\fi
7115    }%
```

The first use full form and the inline full form use the short (long) style.

```
7116    \renewcommand*{\glxtrininlinefullformat}[2]{%
```

```

7117 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7118 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7119 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7120 }%
7121 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7122 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7123 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7124 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7125 }%
7126 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7127 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7128 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7129 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
7130 }%
7131 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7132 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7133 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7134 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
7135 }%
7136 }

```

rt-postfootnote

```
7137 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

shortnolongname

```

7138 \newcommand*{\glxtrshortnolongname}{%
7139 \protect\glsabbrvfont{\the\glsshorttok}%
7140 }

```

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.

```

7141 \newabbreviationstyle{short}%
7142 {%
7143 \renewcommand*{\CustomAbbreviationFields}{%
7144 name={\glxtrshortnolongname},
7145 sort={\the\glsshorttok},
7146 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
7147 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
7148 text={\protect\glsabbrvfont{\the\glsshorttok}},
7149 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
7150 description={\the\glslongtok}}%
7151 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7152 \glssetAttribute{\the\glslabeltok}{regular}{true}}%
7153 }%
7154 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7155 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7156 \renewcommand*{\glstabrvfont}[1]{\glstabrvdefaultfont{##1}}%
7157 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7158 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7159 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7160 \renewcommand*{\glxtrinlinefullformat}[2]{%
7161   \protect\glsfirstabbrvfont{\glssaccessshort{##1}}%
7162   \ifglxtrinsertinside##2\fi}%
7163   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7164   \glxtrparen{\glsfirstlongfont{\glssaccesslong{##1}}}%
7165 }%
7166 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7167   \protect\glsfirstabbrvfont{\glssaccessshortpl{##1}}%
7168   \ifglxtrinsertinside##2\fi}%
7169   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7170   \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
7171 }%
7172 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7173   \protect\glsfirstabbrvfont{\glssaccessshort{##1}}%
7174   \ifglxtrinsertinside##2\fi}%
7175   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7176   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7177 }%
7178 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7179   \protect\glsfirstabbrvfont{\glssaccessshortpl{##1}}%
7180   \ifglxtrinsertinside##2\fi}%
7181   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7182   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7183 }%

```

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.

```

7184 \renewcommand*{\glxtrfullformat}[2]{%
7185   \glsfirstabbrvfont{\glssaccessshort{##1}}\ifglxtrinsertinside##2\fi}%
7186   \ifglxtrinsertinside\else##2\fi
7187 }%
7188 \renewcommand*{\glxtrfullplformat}[2]{%
7189   \glsfirstabbrvfont{\glssaccessshortpl{##1}}\ifglxtrinsertinside##2\fi}%
7190   \ifglxtrinsertinside\else##2\fi
7191 }%
7192 \renewcommand*{\Glsxtrfullformat}[2]{%
7193   \glsfirstabbrvfont{\glssaccessshort{##1}}\ifglxtrinsertinside##2\fi}%
7194   \ifglxtrinsertinside\else##2\fi
7195 }%
7196 \renewcommand*{\Glsxtrfullplformat}[2]{%
7197   \glsfirstabbrvfont{\glssaccessshortpl{##1}}\ifglxtrinsertinside##2\fi}%
7198   \ifglxtrinsertinside\else##2\fi
7199 }%

```

7200 }

Set this as the default style for acronyms:

7201 \setabbreviationstyle[acronym]{short}

short-nolong

7202 \letabbreviationstyle{short-nolong}{short}

rt-nolong-noreg Like short-nolong but doesn't set the regular attribute.

7203 \newabbreviationstyle{short-nolong-noreg}{%

7204 {%

7205 \GlsXtrUseAbbrStyleSetup{short-nolong}%

Unset the regular attribute if it has been set.

7206 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

7207 \glshasattribute{\the\glslabeltok}{regular}%

7208 {%

7209 \glsselattribute{\the\glslabeltok}{regular}{false}%

7210 }%

7211 {}%

7212 }%

7213 }%

7214 {%

7215 \GlsXtrUseAbbrStyleFmts{short-nolong}%

7216 }

trshortdescname

7217 \newcommand*{\glxtrshortdescname}{%

7218 \protect\glsabbrvfont{\the\glsshorttok}%

7219 }

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.

7220 \newabbreviationstyle{short-desc}{%

7221 {%

7222 \renewcommand*{\CustomAbbreviationFields}{%

7223 name={\glxtrshortdescname},

7224 sort={\the\glsshorttok},

7225 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},

7226 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},

7227 text={\protect\glsabbrvfont{\the\glsshorttok}},

7228 plural={\protect\glsabbrvfont{\the\glsshortpltok}},

7229 description={\the\glslongtok}}%

7230 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

7231 \glsselattribute{\the\glslabeltok}{regular}{true}}%

7232 }%

7233 {%

In case the user wants to mix and match font styles, these are redefined here.

```

7234 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
7235 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
7236 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7237 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7238 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

7239 \renewcommand*{\glxtrinlinefullformat}[2]{%
7240   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7241   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7242   \glxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
7243 }%
7244 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7245   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7246   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7247   \glxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
7248 }%
7249 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7250   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7251   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7252   \glxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
7253 }%
7254 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7255   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7256   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7257   \glxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
7258 }%

```

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.

```

7259 \renewcommand*{\glxtrfullformat}[2]{%
7260   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7261   \ifglxtrininsertinside\else##2\fi
7262 }%
7263 \renewcommand*{\glxtrfullplformat}[2]{%
7264   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7265   \ifglxtrininsertinside\else##2\fi
7266 }%
7267 \renewcommand*{\Glsxtrfullformat}[2]{%
7268   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7269   \ifglxtrininsertinside\else##2\fi
7270 }%
7271 \renewcommand*{\Glsxtrfullplformat}[2]{%
7272   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7273   \ifglxtrininsertinside\else##2\fi
7274 }%
7275 }

```

ort-nolong-desc

7276 \letabbreviationstyle{short-nolong-desc}{short-desc}

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

7277 \newabbreviationstyle{short-nolong-desc-noreg}%

7278 {%

7279 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%

Unset the regular attribute if it has been set.

7280 \renewcommand*{\GlsXtrPostNewAbbreviation}{%

7281 \glshasattribute{\the\glslabeltok}{regular}%

7282 {%

7283 \glissetattribute{\the\glslabeltok}{regular}{false}%

7284 }%

7285 {}%

7286 }%

7287 }%

7288 {%

7289 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%

7290 }

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

7291 \newabbreviationstyle{nolong-short}%

7292 {%

7293 \GlsXtrUseAbbrStyleSetup{short-nolong}%

7294 }%

7295 {%

7296 \GlsXtrUseAbbrStyleFmts{short-nolong}%

The inline full form displays the long form followed by the short form in parentheses.

7297 \renewcommand*{\glsxtrinlinefullformat}[2]{%

7298 \protect\glsfirstlongfont{\glssaccesslong{##1}}%

7299 \ifglsxtrininsertinside##2\fi}%

7300 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%

7301 \glsxtrparen{\glsfirstabbrvfont{\glssaccessshort{##1}}}%

7302 }%

7303 \renewcommand*{\glsxtrinlinefullplformat}[2]{%

7304 \protect\glsfirstlongfont{\glssaccesslongpl{##1}}%

7305 \ifglsxtrininsertinside##2\fi}%

7306 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%

7307 \glsxtrparen{\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%

7308 }%

7309 \renewcommand*{\Glsxtrinlinefullformat}[2]{%

7310 \protect\glsfirstlongfont{\glssaccesslong{##1}}%

7311 \ifglsxtrininsertinside##2\fi}%

7312 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}}%

7313 \glsxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%

7314 }%

7315 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%

7316 \protect\glsfirstlongfont{\glssaccesslongpl{##1}}%

```

7317      \ifglxtrinsertinside##2\fi}%
7318      \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7319      \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
7320    }%
7321  }

```

long-short-noreg Like **nolong-short** but doesn't set the regular attribute.

```

7322 \newabbreviationstyle{nolong-short-noreg}%
7323 {%
7324   \GlsXtrUseAbbrStyleSetup{nolong-short}%

   Unset the regular attribute if it has been set.
7325   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7326     \glshasattribute{\the\glslabeltok}{regular}%
7327     {%
7328       \glissetattribute{\the\glslabeltok}{regular}{false}%
7329     }%
7330   }%
7331 }%
7332 }%
7333 {%
7334   \GlsXtrUseAbbrStyleFmts{nolong-short}%
7335 }

```

noshortdescname

```

7336 \newcommand*{\glxtrlongnoshortdescname}{%
7337   \protect\glslongfont{\the\glslongtok}%
7338 }

```

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.

```

7339 \newabbreviationstyle{long-desc}%
7340 {%
7341   \renewcommand*{\CustomAbbreviationFields}{%
7342     name={\glxtrlongnoshortdescname},
7343     sort={\the\glslongtok},
7344     first={\protect\glsfirstlongfont{\the\glslongtok}},
7345     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7346     text={\glslongfont{\the\glslongtok}},
7347     plural={\glslongfont{\the\glslongpltok}}%
7348   }%
7349   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7350     \glissetattribute{\the\glslabeltok}{regular}{true}}%
7351 }%
7352 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7353   \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%

```

```

7354 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
7355 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
7356 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
7357 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

7358 \renewcommand*\glsxtrsubsequentfmt[2]{%
7359   \glslongfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
7360   \ifglsxtrininsertinside \else##2\fi
7361 }%
7362 \renewcommand*\glsxtrsubsequentplfmt[2]{%
7363   \glslongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
7364   \ifglsxtrininsertinside \else##2\fi
7365 }%
7366 \renewcommand*\Glsxtrsubsequentfmt[2]{%
7367   \glslongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
7368   \ifglsxtrininsertinside \else##2\fi
7369 }%
7370 \renewcommand*\Glsxtrsubsequentplfmt[2]{%
7371   \glslongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
7372   \ifglsxtrininsertinside \else##2\fi
7373 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

7374 \renewcommand*\glsxtrinlinefullformat[2]{%
7375   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7376   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7377   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7378 }%
7379 \renewcommand*\glsxtrinlinefullplformat[2]{%
7380   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7381   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7382   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7383 }%
7384 \renewcommand*\Glsxtrinlinefullformat[2]{%
7385   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7386   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7387   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
7388 }%
7389 \renewcommand*\Glsxtrinlinefullplformat[2]{%
7390   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7391   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7392   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
7393 }%

```

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.

```

7394 \renewcommand*\glsxtrfullformat[2]{%
7395   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7396   \ifglsxtrininsertinside\else##2\fi
7397 }%

```

```

7398 \renewcommand*{\glstrfullplformat}[2]{%
7399   \glstrfirstlongfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
7400   \ifglstrinsertinside\else##2\fi
7401 }%
7402 \renewcommand*{\glstrfullformat}[2]{%
7403   \glstrfirstlongfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
7404   \ifglstrinsertinside\else##2\fi
7405 }%
7406 \renewcommand*{\glstrfullplformat}[2]{%
7407   \glstrfirstlongfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
7408   \ifglstrinsertinside\else##2\fi
7409 }%
7410 }

```

`long-noshort-desc` Provide a synonym that matches similar styles.

```

7411 \letabbreviationstyle{long-noshort-desc}{long-desc}

```

`long-noshort-desc-noreg` Like `long-noshort-desc` but doesn't set the regular attribute.

```

7412 \newabbreviationstyle{long-noshort-desc-noreg}%
7413 {%
7414   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
7415   \Unsetthe{regular}%
7416   \glshasattribute{\the\glslabeltok}{regular}%
7417   {%
7418     \glsssetattribute{\the\glslabeltok}{regular}{false}%
7419   }%
7420 }%
7421 }%
7422 }%
7423 {%
7424   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
7425 }

```

`longnoshortname`

```

7426 \newcommand*{\glstrlongnoshortname}{%
7427   \protect\glssabrvfont{\the\glssshorttok}%
7428 }

```

`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.

```

7429 \newabbreviationstyle{long}%
7430 {%
7431   \renewcommand*{\CustomAbbreviationFields}{%
7432     name={\glstrlongnoshortname},
7433     sort={\the\glssshorttok},
7434     first={\protect\glstrfirstlongfont{\the\glslongtok}},

```

```

7435     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
7436     text={\glslongfont{\the\glslongtok}},
7437     plural={\glslongfont{\the\glslongpltok}},%
7438     description={\the\glslongtok}%
7439 }%
7440 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7441   \glssetattribute{\the\glslabeltok}{regular}{true}}%
7442 }%
7443 {%
7444   \GlsXtrUseAbbrStyleFmts{long-desc}%
7445 }

```

`long-noshort` Provide a synonym that matches similar styles.

```
7446 \letabbreviationstyle{long-noshort}{long}
```

`g-noshort-noreg` Like `long-noshort` but doesn't set the regular attribute.

```

7447 \newabbreviationstyle{long-noshort-noreg}%
7448 {%
7449   \GlsXtrUseAbbrStyleSetup{long-noshort}%

```

Unset the regular attribute if it has been set.

```

7450   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7451     \glschasattribute{\the\glslabeltok}{regular}%
7452     {%
7453       \glssetattribute{\the\glslabeltok}{regular}{false}%
7454     }%
7455     {}%
7456   }%
7457 }%
7458 {%
7459   \GlsXtrUseAbbrStyleFmts{long-noshort}%
7460 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glxtrscfont` Maintained for backward-compatibility.

```
7461 \newcommand*{\glxtrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
7462 \newcommand*{\glsabbrvscfont}{\glxtrscfont}
```

`sxtrfirstscfont` Maintained for backward-compatibility.

```
7463 \newcommand*{\sxtrfirstscfont}[1]{\glsabbrvscfont{#1}}
```

`irstabbrvscfont` Added for consistent naming.

```
7464 \newcommand*{\glsfirstabbrvscfont}{\glxtrfirstscfont}
```

and for the default short form suffix:

`\glxtrscsuffix`

```
7465 \newcommand*{\glxtrscsuffix}{\glstextup{\glxtrabbrvpluralsuffix}}
```

`long-short-sc`

```
7466 \newabbreviationstyle{long-short-sc}%
7467 {%
7468   \renewcommand*{\CustomAbbreviationFields}{%
7469     name={\glxtrlongshortname},
7470     sort={\the\glsshorttok},
7471     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7472       \protect\glxtrfullsep{\the\glslabeltok}%
7473       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7474     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7475       \protect\glxtrfullsep{\the\glslabeltok}%
7476       \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7477     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}},%
7478     description={\the\glslongtok}}%
7479   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7480     \glshasattribute{\the\glslabeltok}{regular}%
7481     {%
7482       \glissetattribute{\the\glslabeltok}{regular}{false}%
7483     }%
7484   }%
7485 }%
7486 }%
7487 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
7488 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7489 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7490 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
```

Use the default long fonts.

```
7491 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7492 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
7493 \renewcommand*{\glxtrfullformat}[2]{%
7494   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
7495   \ifglxtrininsertinside\else##2\fi
7496   \glxtrfullsep{##1}%
7497   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshort{##1}}}%
7498 }%
7499 \renewcommand*{\glxtrfullplformat}[2]{%
7500   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
7501   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7502   \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
7503 }%
7504 \renewcommand*{\Glsxtrfullformat}[2]{%
```

```

7505 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7506 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7507 \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
7508 }%
7509 \renewcommand*{\Glsxtrfullplformat}[2]{%
7510 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7511 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7512 \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
7513 }%
7514 }

```

g-short-sc-desc

```

7515 \newabbreviationstyle{long-short-sc-desc}%
7516 {%
7517 \renewcommand*{\CustomAbbreviationFields}{%
7518 name={\glxtrlongshortdescname},
7519 sort={\glxtrlongshortdescsort},%
7520 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
7521 \protect\glxtrfullsep{\the\glslabeltok}%
7522 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
7523 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
7524 \protect\glxtrfullsep{\the\glslabeltok}%
7525 \glxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
7526 text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7527 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7528 }%

```

Unset the regular attribute if it has been set.

```

7529 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7530 \glshasattribute{\the\glslabeltok}{regular}%
7531 {%
7532 \glissetattribute{\the\glslabeltok}{regular}{false}%
7533 }%
7534 {}%
7535 }%
7536 }%
7537 {%

```

As long-short-sc style:

```

7538 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
7539 }

```

Now the short (long) version

```

7540 \newabbreviationstyle{short-sc-long}%
7541 {%
7542 \renewcommand*{\CustomAbbreviationFields}{%
7543 name={\glxtrshortlongname},
7544 sort={\the\glsshorttok},
7545 description={\the\glslongtok},%
7546 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%

```

```

7547 \protect\glxtrfullsep{\the\glslabeltok}%
7548 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
7549 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7550 \protect\glxtrfullsep{\the\glslabeltok}%
7551 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
7552 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

7553 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7554 \glshasattribute{\the\glslabeltok}{regular}%
7555 {%
7556 \glissetattribute{\the\glslabeltok}{regular}{false}%
7557 }%
7558 {}%
7559 }%
7560}%
7561{%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7562 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7563 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7564 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
7565 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7566 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7567 \renewcommand*{\glxtrfullformat}[2]{%
7568 \glsfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7569 \ifglxtrininsertinside\else##2\fi
7570 \glxtrfullsep{##1}%
7571 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
7572}%
7573 \renewcommand*{\glxtrfullplformat}[2]{%
7574 \glsfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7575 \ifglxtrininsertinside\else##2\fi
7576 \glxtrfullsep{##1}%
7577 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
7578}%
7579 \renewcommand*{\Glsxtrfullformat}[2]{%
7580 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7581 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7582 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
7583}%
7584 \renewcommand*{\Glsxtrfullplformat}[2]{%
7585 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7586 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7587 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
7588}%
7589}

```

As before but user provides description

```

7590 \newabbreviationstyle{short-sc-long-desc}%
7591 {%
7592   \renewcommand*{\CustomAbbreviationFields}{%
7593     name={\glxtrshortlongdescname},
7594     sort={\glxtrshortlongdescsort},
7595     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
7596       \protect\glxtrfullsep{\the\glslabeltok}%
7597       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
7598     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7599       \protect\glxtrfullsep{\the\glslabeltok}%
7600       \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
7601     text={\protect\glsabbrvscfont{\the\glsshorttok}},%
7602     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7603   }%

```

Unset the regular attribute if it has been set.

```

7604   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7605     \glshasattribute{\the\glslabeltok}{regular}%
7606     {%
7607       \glissetattribute{\the\glslabeltok}{regular}{false}%
7608     }%
7609   }%
7610 }%
7611 }%
7612 {%

```

As short-sc-long style:

```

7613   \GlsXtrUseAbbrStyleFmts{short-sc-long}%
7614 }

```

short-sc

```

7615 \newabbreviationstyle{short-sc}%
7616 {%
7617   \renewcommand*{\CustomAbbreviationFields}{%
7618     name={\glxtrshortnolongname},
7619     sort={\the\glsshorttok},
7620     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
7621     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
7622     text={\protect\glsabbrvscfont{\the\glsshorttok}},
7623     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
7624     description={\the\glslongtok}}%
7625   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7626     \glissetattribute{\the\glslabeltok}{regular}{true}}%
7627 }%
7628 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7629   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7630   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7631   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%

```

```

7632 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7633 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

7634 \renewcommand*{\glsxtrinlinefullformat}[2]{%
7635   \protect\glsfirstabbrvscfont{\glsaccesssshort{##1}}%
7636   \ifglsxtrinsertinside##2\fi}%
7637   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7638   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7639 }%
7640 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
7641   \protect\glsfirstabbrvscfont{\glsaccesssshortpl{##1}}%
7642   \ifglsxtrinsertinside##2\fi}%
7643   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7644   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7645 }%

7646 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7647   \protect\glsfirstabbrvscfont{\Glsaccesssshort{##1}}%
7648   \ifglsxtrinsertinside##2\fi}%
7649   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7650   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
7651 }%
7652 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7653   \protect\glsfirstabbrvscfont{\Glsaccesssshortpl{##1}}%
7654   \ifglsxtrinsertinside##2\fi}%
7655   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
7656   \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
7657 }%

```

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.

```

7658 \renewcommand*{\glsxtrfullformat}[2]{%
7659   \glsfirstabbrvscfont{\glsaccesssshort{##1}}\ifglsxtrinsertinside##2\fi}%
7660   \ifglsxtrinsertinside\else##2\fi
7661 }%
7662 \renewcommand*{\glsxtrfullplformat}[2]{%
7663   \glsfirstabbrvscfont{\glsaccesssshortpl{##1}}\ifglsxtrinsertinside##2\fi}%
7664   \ifglsxtrinsertinside\else##2\fi
7665 }%
7666 \renewcommand*{\Glsxtrfullformat}[2]{%
7667   \glsfirstabbrvscfont{\Glsaccesssshort{##1}}\ifglsxtrinsertinside##2\fi}%
7668   \ifglsxtrinsertinside\else##2\fi
7669 }%
7670 \renewcommand*{\Glsxtrfullplformat}[2]{%
7671   \glsfirstabbrvscfont{\Glsaccesssshortpl{##1}}\ifglsxtrinsertinside##2\fi}%
7672   \ifglsxtrinsertinside\else##2\fi
7673 }%
7674 }

```

short-sc-nolong

```
7675 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
7676 \newabbreviationstyle{short-sc-desc}{%
7677 {%
7678   \renewcommand*{\CustomAbbreviationFields}{%
7679     name={\glxtrshortdescname},
7680     sort={\the\glsshorttok},
7681     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
7682     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
7683     text={\protect\glsabbrvscfont{\the\glsshorttok}},
7684     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
7685     description={\the\glslongtok}}%
7686   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7687     \glssetattribute{\the\glslabeltok}{regular}{true}}%
7688 }%
7689 }
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
7690 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7691 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
7692 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
7693 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7694 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
7695 \renewcommand*{\glsxtrinlinefullformat}[2]{%
7696   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
7697   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7698   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7699 }%
7700 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
7701   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
7702   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7703   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7704 }%
7705 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7706   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
7707   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7708   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
7709 }%
7710 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7711   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
7712   \ifglsxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7713   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
7714 }
```

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.

```

7715 \renewcommand*{\glxtrfullformat}[2]{%
7716   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7717   \ifglxtrinsertinside\else##2\fi
7718 }%
7719 \renewcommand*{\glxtrfullplformat}[2]{%
7720   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7721   \ifglxtrinsertinside\else##2\fi
7722 }%
7723 \renewcommand*{\Glsxtrfullformat}[2]{%
7724   \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
7725   \ifglxtrinsertinside\else##2\fi
7726 }%
7727 \renewcommand*{\Glsxtrfullplformat}[2]{%
7728   \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7729   \ifglxtrinsertinside\else##2\fi
7730 }%
7731 }

```

-sc-nolong-desc

```

7732 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}

```

nolong-short-sc

```

7733 \newabbreviationstyle{nolong-short-sc}%
7734 {%
7735   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
7736 }%
7737 {%
7738   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

7739 \renewcommand*{\glxtrinlinefullformat}[2]{%
7740   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
7741     \ifglxtrinsertinside##2\fi}%
7742   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7743   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7744 }%
7745 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7746   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
7747     \ifglxtrinsertinside##2\fi}%
7748   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7749   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7750 }%
7751 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7752   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
7753     \ifglxtrinsertinside##2\fi}%
7754   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7755   \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
7756 }%
7757 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7758   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%

```

```

7759     \ifglxtrinsertinside##2\fi}%
7760     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7761     \glxtrparen{\glsfirstabbrvscfont{\glssaccessshortpl{##1}}}%
7762 }%
7763 }

```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`.

```

7764 \newabbreviationstyle{long-noshort-sc}%
7765 {%
7766   \renewcommand*{\CustomAbbreviationFields}{%
7767     name={\glxtrlongnoshortname},
7768     sort={\the\glssshorttok},
7769     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
7770     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
7771     text={\protect\glslongdefaultfont{\the\glslongtok}},
7772     plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
7773     description={\the\glslongtok}%
7774   }%
7775   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7776     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
7777 }%
7778 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7779   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7780   \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
7781   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
7782   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7783   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

7784   \renewcommand*{\glxtrsubsequentfmt}[2]{%
7785     \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7786     \ifglxtrinsertinside \else##2\fi
7787   }%
7788   \renewcommand*{\glxtrsubsequentplfmt}[2]{%
7789     \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7790     \ifglxtrinsertinside \else##2\fi
7791   }%
7792   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
7793     \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
7794     \ifglxtrinsertinside \else##2\fi
7795   }%
7796   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
7797     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
7798     \ifglxtrinsertinside \else##2\fi
7799   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

7800 \renewcommand*{\glxtrinlinefullformat}[2]{%
7801   \glsfirslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7802   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7803   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7804 }%
7805 \renewcommand*{\glxtrinlinefullplformat}[2]{%
7806   \glsfirslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7807   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7808   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7809 }%
7810 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
7811   \glsfirslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7812   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7813   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
7814 }%
7815 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
7816   \glsfirslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7817   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7818   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
7819 }%

```

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.

```

7820 \renewcommand*{\glxtrfullformat}[2]{%
7821   \glsfirslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7822   \ifglxtrinsertinside\else##2\fi
7823 }%
7824 \renewcommand*{\glxtrfullplformat}[2]{%
7825   \glsfirslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7826   \ifglxtrinsertinside\else##2\fi
7827 }%
7828 \renewcommand*{\Glsxtrfullformat}[2]{%
7829   \glsfirslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7830   \ifglxtrinsertinside\else##2\fi
7831 }%
7832 \renewcommand*{\Glsxtrfullplformat}[2]{%
7833   \glsfirslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7834   \ifglxtrinsertinside\else##2\fi
7835 }%
7836 }

```

long-sc Backward compatibility:

```

7837 \@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`.

```

7838 \newabbreviationstyle{long-noshort-sc-desc}%
7839 {%
7840   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%

```

7841 }%

7842 {%

Use smallcaps and adjust the plural suffix to revert to upright.

7843 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%

7844 \renewcommand*{\glabbrvfont}[1]{\glabbrvscfont{##1}}%

7845 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvscfont{##1}}%

7846 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%

7847 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

The format for subsequent use (not used when the regular attribute is set).

7848 \renewcommand*{\glxtrsubsequentfmt}[2]{%

7849 \glslongdefaultfont{\glaccesslong{##1}\ifglxtrininsertinside ##2\fi}%

7850 \ifglxtrininsertinside \else##2\fi

7851 }%

7852 \renewcommand*{\glxtrsubsequentplfmt}[2]{%

7853 \glslongdefaultfont{\glaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%

7854 \ifglxtrininsertinside \else##2\fi

7855 }%

7856 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%

7857 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%

7858 \ifglxtrininsertinside \else##2\fi

7859 }%

7860 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%

7861 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%

7862 \ifglxtrininsertinside \else##2\fi

7863 }%

The inline full form displays the long format followed by the short form in parentheses.

7864 \renewcommand*{\glxtrinlinefullformat}[2]{%

7865 \glfirstlongdefaultfont{\glaccesslong{##1}\ifglxtrininsertinside##2\fi}%

7866 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

7867 \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshort{##1}}}%

7868 }%

7869 \renewcommand*{\glxtrinlinefullplformat}[2]{%

7870 \glfirstlongdefaultfont{\glaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%

7871 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

7872 \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshortpl{##1}}}%

7873 }%

7874 \renewcommand*{\Glsxtrinlinefullformat}[2]{%

7875 \glfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%

7876 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

7877 \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshort{##1}}}%

7878 }%

7879 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%

7880 \glfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%

7881 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

7882 \glxtrparen{\protect\glfirstabbrvscfont{\glaccessshortpl{##1}}}%

7883 }%

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.

```
7884 \renewcommand*{\glxstrfullformat}[2]{%
7885   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7886   \ifglxtrinsertinside\else##2\fi
7887 }%
7888 \renewcommand*{\glxstrfullplformat}[2]{%
7889   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7890   \ifglxtrinsertinside\else##2\fi
7891 }%
7892 \renewcommand*{\Glsxtrfullformat}[2]{%
7893   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
7894   \ifglxtrinsertinside\else##2\fi
7895 }%
7896 \renewcommand*{\Glsxtrfullplformat}[2]{%
7897   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
7898   \ifglxtrinsertinside\else##2\fi
7899 }%
7900 }
```

long-desc-sc Backward compatibility:

```
7901 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
7902 \newabbreviationstyle{short-sc-footnote}%
7903 {%
7904   \renewcommand*{\CustomAbbreviationFields}{%
7905     name={\glxstrfootnotename},
7906     sort={\the\glsshorttok},
7907     description={\the\glslongtok},%
7908     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
7909       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
7910       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
7911     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
7912       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
7913       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
7914     plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
7915 }
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
7915 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7916   \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
7917   \glshasattribute{\the\glslabeltok}{regular}%
7918   {%
7919     \glsssetattribute{\the\glslabeltok}{regular}{false}%
7920   }%
7921   {}%
7922 }%
7923 }%
7924 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```

7925 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
7926 \renewcommand*{\glabbrvfont}[1]{\glabbrvscfont{##1}}%
7927 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvscfont{##1}}%
7928 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongfootnotefont{##1}}%
7929 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

7930 \renewcommand*{\glxtrfullformat}[2]{%
7931   \glfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7932   \ifglxtrininsertinside\else##2\fi
7933   \protect\glxtrabbrvfootnote{##1}%
7934   {\glfirstlongfootnotefont{\glaccesslong{##1}}}%
7935 }%
7936 \renewcommand*{\glxtrfullplformat}[2]{%
7937   \glfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7938   \ifglxtrininsertinside\else##2\fi
7939   \protect\glxtrabbrvfootnote{##1}%
7940   {\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
7941 }%
7942 \renewcommand*{\Glsxtrfullformat}[2]{%
7943   \glfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7944   \ifglxtrininsertinside\else##2\fi
7945   \protect\glxtrabbrvfootnote{##1}%
7946   {\glfirstlongfootnotefont{\glaccesslong{##1}}}%
7947 }%
7948 \renewcommand*{\Glsxtrfullplformat}[2]{%
7949   \glfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7950   \ifglxtrininsertinside\else##2\fi
7951   \protect\glxtrabbrvfootnote{##1}%
7952   {\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
7953 }%

```

The first use full form and the inline full form use the short (long) style.

```

7954 \renewcommand*{\glxtrininlinefullformat}[2]{%
7955   \glfirstabbrvscfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7956   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7957   \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%
7958 }%
7959 \renewcommand*{\glxtrininlinefullplformat}[2]{%
7960   \glfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
7961   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7962   \glxtrparen{\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
7963 }%
7964 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
7965   \glfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
7966   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
7967   \glxtrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%
7968 }%
7969 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%

```

```

7970 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
7971 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
7972 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
7973 }%
7974 }

```

footnote-sc Backward compatibility:

```

7975 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```

sc-postfootnote

```

7976 \newabbreviationstyle{short-sc-postfootnote}%
7977 {%
7978 \renewcommand*{\CustomAbbreviationFields}{%
7979 name={\glxtrfootnotename},
7980 sort={\the\glsshorttok},
7981 description={\the\glslongtok},%
7982 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
7983 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
7984 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.

```

7985 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7986 \csdef{glxtrpostlink\glscategorylabel}{%
7987 \glxtrifwasfirstuse
7988 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

7989 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
7990 {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
7991 }%
7992 {}%
7993 }%
7994 \glshasattribute{\the\glslabeltok}{regular}%
7995 {%
7996 \glissetattribute{\the\glslabeltok}{regular}{false}%
7997 }%
7998 {}%
7999 }%

```

The footnote needs to be suppressed in the inline form, so \glxtrfull must set the first use switch off.

```

8000 \renewcommand*{\glxtrsetupfulldefs}{%
8001 \let\glxtrifwasfirstuse\@secondoftwo
8002 }%
8003 }%
8004 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8005 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
8006 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvscfont{##1}}%
8007 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvscfont{##1}}%
8008 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongfootnotefont{##1}}%
8009 \renewcommand*{\glsslongfont}[1]{\glsslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8010 \renewcommand*{\glxtrfullformat}[2]{%
8011   \glssfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8012   \ifglxtrininsertinside\else##2\fi
8013 }%
8014 \renewcommand*{\glxtrfullplformat}[2]{%
8015   \glssfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8016   \ifglxtrininsertinside\else##2\fi
8017 }%
8018 \renewcommand*{\Glsxtrfullformat}[2]{%
8019   \glssfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8020   \ifglxtrininsertinside\else##2\fi
8021 }%
8022 \renewcommand*{\Glsxtrfullplformat}[2]{%
8023   \glssfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8024   \ifglxtrininsertinside\else##2\fi
8025 }%

```

The first use full form and the inline full form use the short (long) style.

```

8026 \renewcommand*{\glxtrinlinefullformat}[2]{%
8027   \glssfirstabbrvscfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8028   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8029   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
8030 }%
8031 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8032   \glssfirstabbrvscfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8033   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8034   \glxtrparen{\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
8035 }%
8036 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8037   \glssfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8038   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8039   \glxtrparen{\glssfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8040 }%
8041 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8042   \glssfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8043   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8044   \glxtrparen{\glssfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8045 }%
8046 }

```

postfootnote-sc Backward compatibility:

```

8047 \@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.
8048 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

\glsabrvsmfont   Added for consistent naming.
8049 \newcommand*{\glsabrvsmfont}{\glxtrsmfont}

sxtrfirstsmfont  Maintained for backward compatibility.
8050 \newcommand*{\glxtrfirstsmfont}[1]{\glsabrvsmfont{#1}}

irstabrvsmfont   Added for consistent naming.
8051 \newcommand*{\glfirstabrvsmfont}{\glxtrfirstsmfont}

                and for the default short form suffix:

\glxtrsmsuffix
8052 \newcommand*{\glxtrsmsuffix}{\glxtrabrvpluralsuffix}

long-short-sm
8053 \newabbreviationstyle{long-short-sm}%
8054 {%
8055   \renewcommand*{\CustomAbbreviationFields}{%
8056     name={\glxtrlongshortname},
8057     sort={\the\glsshorttok},
8058     first={\protect\glfirstlongdefaultfont{\the\glslongtok}%
8059       \protect\glxtrfullsep{\the\glslabeltok}%
8060       \glxtrparen{\protect\glfirstabrvsmfont{\the\glsshorttok}}},%
8061     firstplural={\protect\glfirstlongdefaultfont{\the\glslongpltok}%
8062       \protect\glxtrfullsep{\the\glslabeltok}%
8063       \glxtrparen{\protect\glfirstabrvsmfont{\the\glsshortpltok}}},%
8064     plural={\protect\glsabrvsmfont{\the\glsshortpltok}}},%
8065     description={\the\glslongtok}}%
8066   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8067     \glshasattribute{\the\glslabeltok}{regular}%
8068     {%
8069       \glissetattribute{\the\glslabeltok}{regular}{false}%
8070     }%
8071   }%
8072 }%
8073 }%
8074 {%
8075   \renewcommand*{\glsabrvfont}[1]{\glsabrvsmfont{##1}}%
8076   \renewcommand*{\glfirstabrvfont}[1]{\glfirstabrvsmfont{##1}}%
8077   \renewcommand*{\abrvpluralsuffix}{\protect\glxtrsmsuffix}%
```

Use the default long fonts.

```
8078 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8079 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8080 \renewcommand*{\glsxtrfullformat}[2]{%
8081   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8082   \ifglsxtrinsertinside\else##2\fi
8083   \glsxtrfullsep{##1}%
8084   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8085 }%
8086 \renewcommand*{\glsxtrfullplformat}[2]{%
8087   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8088   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8089   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8090 }%
8091 \renewcommand*{\Glsxtrfullformat}[2]{%
8092   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8093   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8094   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
8095 }%
8096 \renewcommand*{\Glsxtrfullplformat}[2]{%
8097   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8098   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8099   \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
8100 }%
8101 }
```

g-short-sm-desc

```
8102 \newabbreviationstyle{long-short-sm-desc}%
8103 {%
8104   \renewcommand*{\CustomAbbreviationFields}{%
8105     name={\glsxtrlongshortdescname},
8106     sort={\glsxtrlongshortdescsort},%
8107     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8108       \protect\glsxtrfullsep{\the\glslabeltok}%
8109       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
8110     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8111       \protect\glsxtrfullsep{\the\glslabeltok}%
8112       \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
8113     text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
8114     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
8115   }%
```

Unset the regular attribute if it has been set.

```
8116 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8117   \glshasattribute{\the\glslabeltok}{regular}%
8118   {%
8119     \glssetattribute{\the\glslabeltok}{regular}{false}%
8120   }%
```

```

8121 {}%
8122 }%
8123 }%
8124 {%

```

As long-short-sm style:

```

8125 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
8126 }

```

short-sm-long Now the short (long) version

```

8127 \newabbreviationstyle{short-sm-long}%
8128 {%
8129 \renewcommand*{\CustomAbbreviationFields}{%
8130 name={\glxtrshortlongname},
8131 sort={\the\glsshorttok},
8132 description={\the\gslongtok},%
8133 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8134 \protect\glxtrfullsep{\the\glslabeltok}%
8135 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongtok}}},%
8136 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8137 \protect\glxtrfullsep{\the\glslabeltok}%
8138 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\gslongpltok}}},%
8139 plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8140 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8141 \glshasattribute{\the\glslabeltok}{regular}%
8142 {%
8143 \glissetattribute{\the\glslabeltok}{regular}{false}%
8144 }%
8145 {}%
8146 }%
8147 }%
8148 {%
8149 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8150 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8151 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
8152 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8153 \renewcommand*\gslongfont[1]{\gslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8154 \renewcommand*\glxtrfullformat[2]{%
8155 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8156 \ifglxtrininsertinside\else##2\fi
8157 \glxtrfullsep{##1}%
8158 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8159 }%
8160 \renewcommand*\glxtrfullplformat[2]{%
8161 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8162 \ifglxtrininsertinside\else##2\fi

```

```

8163 \glxtrfullsep{##1}%
8164 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8165 }%
8166 \renewcommand*{\Glsxtrfullformat}[2]{%
8167 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8168 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8169 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8170 }%
8171 \renewcommand*{\Glsxtrfullplformat}[2]{%
8172 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8173 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8174 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8175 }%
8176 }

```

rt-sm-long-desc As before but user provides description

```

8177 \newabbreviationstyle{short-sm-long-desc}%
8178 {%
8179 \renewcommand*{\CustomAbbreviationFields}{%
8180 name={\glxtrshortlongdescname},
8181 sort={\glxtrshortlongdescsort},
8182 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8183 \protect\glxtrfullsep{\the\glslabeltok}%
8184 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8185 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8186 \protect\glxtrfullsep{\the\glslabeltok}%
8187 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8188 text={\protect\glssabbrvsmfont{\the\glsshorttok}},%
8189 plural={\protect\glssabbrvsmfont{\the\glsshortpltok}}}%
8190 }%

```

Unset the regular attribute if it has been set.

```

8191 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8192 \glshasattribute{\the\glslabeltok}{regular}%
8193 {%
8194 \glissetattribute{\the\glslabeltok}{regular}{false}%
8195 }%
8196 {}%
8197 }%
8198 }%
8199 {%

```

As short-sm-long style:

```

8200 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
8201 }

```

short-sm

```

8202 \newabbreviationstyle{short-sm}%
8203 {%
8204 \renewcommand*{\CustomAbbreviationFields}{%

```

```

8205     name={\glxtrshortnolongname},
8206     sort={\the\glsshorttok},
8207     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8208     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8209     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8210     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8211     description={\the\glslongtok}}%
8212 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8213   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8214 }%
8215 {%
8216 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8217 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8218 \renewcommand*\abbrvpluralsuffix{\protect\glxtrmsuffix}%
8219 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8220 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8221 \renewcommand*\glxtrinlinefullformat[2]{%
8222   \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}%
8223   \ifglxtrininsertinside##2\fi}%
8224 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8225 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8226 }%
8227 \renewcommand*\glxtrinlinefullplformat[2]{%
8228   \protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}%
8229   \ifglxtrininsertinside##2\fi}%
8230 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8231 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8232 }%
8233 \renewcommand*\Glsxtrinlinefullformat[2]{%
8234   \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}}%
8235   \ifglxtrininsertinside##2\fi}%
8236 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8237 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8238 }%
8239 \renewcommand*\Glsxtrinlinefullplformat[2]{%
8240   \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}%
8241   \ifglxtrininsertinside##2\fi}%
8242 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8243 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8244 }%

```

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.

```

8245 \renewcommand*\glxtrfullformat[2]{%
8246   \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8247   \ifglxtrininsertinside\else##2\fi
8248 }%

```

```

8249 \renewcommand*{\glxtrfullplformat}[2]{%
8250   \glsfirstabbrvsmfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8251   \ifglxtrininsertinside\else##2\fi
8252 }%
8253 \renewcommand*{\Glsxtrfullformat}[2]{%
8254   \glsfirstabbrvsmfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8255   \ifglxtrininsertinside\else##2\fi
8256 }%
8257 \renewcommand*{\Glsxtrfullplformat}[2]{%
8258   \glsfirstabbrvsmfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8259   \ifglxtrininsertinside\else##2\fi
8260 }%
8261 }

```

short-sm-nolong

```

8262 \letabbreviationstyle{short-sm-nolong}{short-sm}

```

short-sm-desc

```

8263 \newabbreviationstyle{short-sm-desc}%
8264 {%
8265   \renewcommand*{\CustomAbbreviationFields}{%
8266     name={\glxtrshortdescname},
8267     sort={\the\glsshorttok},
8268     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
8269     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
8270     text={\protect\glsabbrvsmfont{\the\glsshorttok}},
8271     plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
8272     description={\the\glslongtok}}%
8273   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8274     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8275 }%
8276 {%
8277   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
8278   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
8279   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrrmsuffix}%
8280   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8281   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8282 \renewcommand*{\glxtrininlinefullformat}[2]{%
8283   \glsfirstabbrvsmfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
8284   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8285   \glxtrparen{\glsfirstlongdefaultfont{\glaccesslong{##1}}}%
8286 }%
8287 \renewcommand*{\glxtrininlinefullplformat}[2]{%
8288   \glsfirstabbrvsmfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
8289   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8290   \glxtrparen{\glsfirstlongdefaultfont{\glaccesslongpl{##1}}}%
8291 }%
8292 \renewcommand*{\Glsxtrininlinefullformat}[2]{%

```

```

8293 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8294 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8295 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8296 }%
8297 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8298 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8299 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8300 \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8301 }%

```

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.

```

8302 \renewcommand*{\glxtrfullformat}[2]{%
8303 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8304 \ifglxtrinsertinside\else##2\fi
8305 }%
8306 \renewcommand*{\glxtrfullplformat}[2]{%
8307 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8308 \ifglxtrinsertinside\else##2\fi
8309 }%
8310 \renewcommand*{\Glsxtrfullformat}[2]{%
8311 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8312 \ifglxtrinsertinside\else##2\fi
8313 }%
8314 \renewcommand*{\Glsxtrfullplformat}[2]{%
8315 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8316 \ifglxtrinsertinside\else##2\fi
8317 }%
8318 }

```

-sm-nolong-desc

```

8319 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

8320 \newabbreviationstyle{nolong-short-sm}%
8321 {%
8322 \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
8323 }%
8324 {%
8325 \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8326 \renewcommand*{\glxtrinlinefullformat}[2]{%
8327 \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8328 \ifglxtrinsertinside##2\fi}%
8329 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8330 \glxtrparen{\glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
8331 }%
8332 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8333 \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%

```

```

8334     \ifglxtrinsertinside##2\fi}%
8335     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8336     \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8337 }%
8338 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8339     \protect\glsfirstlongdefaultfont{\glssaccesslong{##1}%
8340         \ifglxtrinsertinside##2\fi}%
8341     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8342     \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%
8343 }%
8344 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8345     \protect\glsfirstlongdefaultfont{\glssaccesslongpl{##1}%
8346         \ifglxtrinsertinside##2\fi}%
8347     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8348     \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%
8349 }%
8350 }

```

`long-noshort-sm` The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

8351 \newabbreviationstyle{long-noshort-sm}%
8352 {%
8353     \renewcommand*{\CustomAbbreviationFields}{%
8354         name={\glxtrlongnoshortname},
8355         sort={\the\glsshorttok},
8356         first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8357         firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8358         text={\protect\glslongdefaultfont{\the\glslongtok}},
8359         plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
8360         description={\the\glslongtok}%
8361     }%
8362     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8363         \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
8364 }%
8365 {%
8366     \renewcommand*\glssabbrvfont[1]{\glssabbrvsmfont{##1}}%
8367     \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8368     \renewcommand*\glssabbrvpluralsuffix{\protect\glssabbrvsmfont{##1}}%
8369     \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8370     \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8371 \renewcommand*{\glssxtrsubsequentfmt}[2]{%
8372     \glslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8373     \ifglxtrinsertinside \else##2\fi
8374 }%
8375 \renewcommand*{\glssxtrsubsequentplfmt}[2]{%
8376     \glslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8377     \ifglxtrinsertinside \else##2\fi
8378 }%

```

```

8379 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
8380   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
8381   \ifglxtrinsertinside \else##2\fi
8382 }%
8383 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
8384   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
8385   \ifglxtrinsertinside \else##2\fi
8386 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8387 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8388   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8389   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8390   \Glsxtrparen{\protect\Glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
8391 }%
8392 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8393   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8394   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8395   \Glsxtrparen{\protect\Glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8396 }%
8397 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8398   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8399   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8400   \Glsxtrparen{\protect\Glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
8401 }%
8402 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8403   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8404   \ifglxtrinsertinside\else##2\fi\Glsxtrfullsep{##1}%
8405   \Glsxtrparen{\protect\Glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8406 }%

```

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.

```

8407 \renewcommand*{\Glsxtrfullformat}[2]{%
8408   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8409   \ifglxtrinsertinside\else##2\fi
8410 }%
8411 \renewcommand*{\Glsxtrfullplformat}[2]{%
8412   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8413   \ifglxtrinsertinside\else##2\fi
8414 }%
8415 \renewcommand*{\Glsxtrfullformat}[2]{%
8416   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8417   \ifglxtrinsertinside\else##2\fi
8418 }%
8419 \renewcommand*{\Glsxtrfullplformat}[2]{%
8420   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8421   \ifglxtrinsertinside\else##2\fi
8422 }%
8423 }

```

long-sm Backward compatibility:

```
8424 \@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.

```
8425 \newabbreviationstyle{long-noshort-sm-desc}%
```

```
8426 {%
```

```
8427 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
```

```
8428 }%
```

```
8429 {%
```

```
8430 \renewcommand*\glssabrvfont[1]{\glssabrvsmfont{##1}}%
```

```
8431 \renewcommand*\glssfirstabrvfont[1]{\glssfirstabrvsmfont{##1}}%
```

```
8432 \renewcommand*\abbrvpluralsuffix{\protect\glsxtrsmsuffix}%
```

```
8433 \renewcommand*\glssfirstlongfont[1]{\glssfirstlongdefaultfont{##1}}%
```

```
8434 \renewcommand*\glsslongfont[1]{\glsslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
8435 \renewcommand*\glsxtrsubsequentfmt}[2]{%
```

```
8436 \glsslongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
8437 \ifglsxtrininsertinside \else##2\fi
```

```
8438 }%
```

```
8439 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
```

```
8440 \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
8441 \ifglsxtrininsertinside \else##2\fi
```

```
8442 }%
```

```
8443 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
```

```
8444 \glsslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
8445 \ifglsxtrininsertinside \else##2\fi
```

```
8446 }%
```

```
8447 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
```

```
8448 \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
```

```
8449 \ifglsxtrininsertinside \else##2\fi
```

```
8450 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
8451 \renewcommand*\glsxtrinlinefullformat}[2]{%
```

```
8452 \glssfirstlongdefaultfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
```

```
8453 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
8454 \glsxtrparen{\protect\glssfirstabrvsmfont{\glssaccessshort{##1}}}%
```

```
8455 }%
```

```
8456 \renewcommand*\glsxtrinlinefullplformat}[2]{%
```

```
8457 \glssfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
```

```
8458 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
8459 \glsxtrparen{\protect\glssfirstabrvsmfont{\glssaccessshortpl{##1}}}%
```

```
8460 }%
```

```
8461 \renewcommand*\Glsxtrinlinefullformat}[2]{%
```

```
8462 \glssfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
```

```
8463 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
8464 \glsxtrparen{\protect\glssfirstabrvsmfont{\glssaccessshort{##1}}}%
```

```
8465 }%
```

```

8466 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8467   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8468   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8469   \glxtrparen{\protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
8470 }%

```

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.

```

8471 \renewcommand*{\glxtrfullformat}[2]{%
8472   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8473   \ifglxtrinsertinside\else##2\fi
8474 }%
8475 \renewcommand*{\glxtrfullplformat}[2]{%
8476   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8477   \ifglxtrinsertinside\else##2\fi
8478 }%
8479 \renewcommand*{\Glsxtrfullformat}[2]{%
8480   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8481   \ifglxtrinsertinside\else##2\fi
8482 }%
8483 \renewcommand*{\Glsxtrfullplformat}[2]{%
8484   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8485   \ifglxtrinsertinside\else##2\fi
8486 }%
8487 }

```

long-desc-sm Backward compatibility:

```

8488 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

8489 \newabbreviationstyle{short-sm-footnote}%
8490 {%
8491   \renewcommand*{\CustomAbbreviationFields}{%
8492     name={\glxtrfootnotename},
8493     sort={\the\glsshorttok},
8494     description={\the\glslongtok},%
8495     first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
8496       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8497       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8498     firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
8499       \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8500       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8501     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.

```

8502 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8503   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8504   \glsattribute{\the\glslabeltok}{regular}%
8505 }%

```

```

8506      \glsetattribute{\the\glslabeltok}{regular}{false}%
8507      }%
8508      {}%
8509      }%
8510 }%
8511 {%
8512 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8513 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8514 \renewcommand*\abbrvpluralsuffix{\protect\glsxrsmssuffix}%
8515 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8516 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8517 \renewcommand*\glsxtrfullformat[2]{%
8518   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8519   \ifglsxtrininsertinside\else##2\fi
8520   \protect\glsxtrabbrvfootnote{##1}%
8521   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8522 }%
8523 \renewcommand*\glsxtrfullplformat[2]{%
8524   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8525   \ifglsxtrininsertinside\else##2\fi
8526   \protect\glsxtrabbrvfootnote{##1}%
8527   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8528 }%
8529 \renewcommand*\Glsxtrfullformat[2]{%
8530   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8531   \ifglsxtrininsertinside\else##2\fi
8532   \protect\glsxtrabbrvfootnote{##1}%
8533   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8534 }%
8535 \renewcommand*\Glsxtrfullplformat[2]{%
8536   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8537   \ifglsxtrininsertinside\else##2\fi
8538   \protect\glsxtrabbrvfootnote{##1}%
8539   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8540 }%

```

The first use full form and the inline full form use the short (long) style.

```

8541 \renewcommand*\glsxtrinlinefullformat[2]{%
8542   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8543   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8544   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8545 }%
8546 \renewcommand*\glsxtrinlinefullplformat[2]{%
8547   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8548   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8549   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8550 }%
8551 \renewcommand*\Glsxtrinlinefullformat[2]{%

```

```

8552 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8553 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8554 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8555 }%
8556 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8557 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8558 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8559 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8560 }%
8561 }

```

footnote-sm Backward compatibility:

```

8562 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

8563 \newabbreviationstyle{short-sm-postfootnote}%
8564 {%
8565 \renewcommand*{\CustomAbbreviationFields}{%
8566 name={\glxtrfootnotename},
8567 sort={\the\glsshorttok},
8568 description={\the\glslongtok},%
8569 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
8570 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
8571 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.

```

8572 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8573 \csdef{glxtrpostlink\glscategorylabel}{%
8574 \glxtrifwasfirstuse
8575 }%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8576 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8577 {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}%
8578 }%
8579 {}%
8580 }%
8581 \glshasattribute{\the\glslabeltok}{regular}%
8582 {%
8583 \glissetattribute{\the\glslabeltok}{regular}{false}%
8584 }%
8585 {}%
8586 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

8587 \renewcommand*{\glxtrsetupfulldefs}{%
8588 \let\glxtrifwasfirstuse\@secondoftwo

```

```

8589 }%
8590 }%
8591 {%
8592 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
8593 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
8594 \renewcommand*\abbrvpluralsuffix{\protect\glsxrtrmsuffix}%
8595 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
8596 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8597 \renewcommand*\glsxtrfullformat[2]{%
8598   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8599   \ifglsxtrininsertinside\else##2\fi
8600 }%
8601 \renewcommand*\glsxtrfullplformat[2]{%
8602   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8603   \ifglsxtrininsertinside\else##2\fi
8604 }%
8605 \renewcommand*\Glsxtrfullformat[2]{%
8606   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8607   \ifglsxtrininsertinside\else##2\fi
8608 }%
8609 \renewcommand*\Glsxtrfullplformat[2]{%
8610   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8611   \ifglsxtrininsertinside\else##2\fi
8612 }%

```

The first use full form and the inline full form use the short (long) style.

```

8613 \renewcommand*\glsxtrinlinefullformat[2]{%
8614   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8615   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8616   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8617 }%
8618 \renewcommand*\glsxtrinlinefullplformat[2]{%
8619   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8620   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8621   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8622 }%
8623 \renewcommand*\Glsxtrinlinefullformat[2]{%
8624   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8625   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8626   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8627 }%
8628 \renewcommand*\Glsxtrinlinefullplformat[2]{%
8629   \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8630   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8631   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8632 }%
8633 }

```

postfootnote-sm Backward compatibility:
8634 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

1.7.5 Predefined Styles (Emphasized)

These styles use \emph for the short form.

\glsabbrvemfont
8635 \newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%

irstabbrvemfont
8636 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%

The default short form suffix:

\glxtremsuffix
8637 \newcommand*{\glxtremsuffix}{\glxtrabbrvpluralsuffix}

firstlongemfont Only used by the “long-em” styles.
8638 \newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{#1}}%

\glslongemfont Only used by the “long-em” styles.
8639 \newcommand*{\glslongemfont}[1]{\emph{#1}}%

long-short-em The long form is just set in the default long font.
8640 \newabbreviationstyle{long-short-em}%
8641 {%
8642 \renewcommand*{\CustomAbbreviationFields}{%
8643 name={\glxtrlongshortname},
8644 sort={\the\glsshorttok},
8645 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}}%
8646 \protect\glxtrfullsep{\the\glslabeltok}}%
8647 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
8648 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}}%
8649 \protect\glxtrfullsep{\the\glslabeltok}}%
8650 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
8651 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
8652 description={\the\glslongtok}}%
8653 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8654 \glshasattribute{\the\glslabeltok}{regular}}%
8655 {%
8656 \glssetattribute{\the\glslabeltok}{regular}{false}}%
8657 }%
8658 }%
8659 }%
8660 }%
8661 {%
8662 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
8663 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
8664 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}}%

Use the default long fonts.

```
8665 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8666 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8667 \renewcommand*{\glsxtrfullformat}[2]{%
8668   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8669   \ifglsxtrinsertinside\else##2\fi
8670   \glsxtrfullsep{##1}%
8671   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
8672 }%
8673 \renewcommand*{\glsxtrfullplformat}[2]{%
8674   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8675   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8676   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
8677 }%
8678 \renewcommand*{\Glsxtrfullformat}[2]{%
8679   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8680   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8681   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
8682 }%
8683 \renewcommand*{\Glsxtrfullplformat}[2]{%
8684   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8685   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8686   \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
8687 }%
8688 }
```

g-short-em-desc

```
8689 \newabbreviationstyle{long-short-em-desc}%
8690 {%
8691   \renewcommand*{\CustomAbbreviationFields}{%
8692     name={\glsxtrlongshortdescname},
8693     sort={\glsxtrlongshortdescsort},%
8694     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8695       \protect\glsxtrfullsep{\the\glslabeltok}%
8696       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
8697     firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8698       \protect\glsxtrfullsep{\the\glslabeltok}%
8699       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
8700     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
8701     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
8702   }%
```

Unset the regular attribute if it has been set.

```
8703 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8704   \glshasattribute{\the\glslabeltok}{regular}%
8705   {%
8706     \glssetattribute{\the\glslabeltok}{regular}{false}%
8707   }%
```

```

8708     {}%
8709   }%
8710 }%
8711 {%

```

As long-short-em style:

```

8712   \GlsXtrUseAbbrStyleFmts{long-short-em}%
8713 }

```

ong-em-short-em

```

8714 \newabbreviationstyle{long-em-short-em}%
8715 {%

```

\glslongemfont is used in the description since \glsdesc doesn't set the style.

```

8716   \renewcommand*{\CustomAbbreviationFields}{%
8717     name={\glsxtrlongshortname},
8718     sort={\the\glsshorttok},
8719     first={\protect\glsfirstlongemfont{\the\glslongtok}%
8720       \protect\glsxtrfullsep{\the\glslabeltok}%
8721       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
8722     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
8723       \protect\glsxtrfullsep{\the\glslabeltok}%
8724       \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%

8725     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}},%
8726     description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

8727   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8728     \glshasattribute{\the\glslabeltok}{regular}%
8729     {%
8730       \glissetattribute{\the\glslabeltok}{regular}{false}%
8731       }%
8732     {}%
8733   }%
8734 }%
8735 {%
8736   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
8737   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
8738   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
8739   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
8740   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8741   \renewcommand*{\glsxtrfullformat}[2]{%
8742     \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8743     \ifglsxtrininsertinside\else##2\fi
8744     \glsxtrfullsep{##1}%
8745     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
8746   }%
8747   \renewcommand*{\glsxtrfullplformat}[2]{%

```

```

8748 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8749 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8750 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
8751 }%
8752 \renewcommand*{\Glsxtrfullformat}[2]{%
8753 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
8754 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8755 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
8756 }%
8757 \renewcommand*{\Glsxtrfullplformat}[2]{%
8758 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
8759 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8760 \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
8761 }%
8762 }

```

m-short-em-desc

```

8763 \newabbreviationstyle{long-em-short-em-desc}%
8764 {%
8765 \renewcommand*{\CustomAbbreviationFields}{%
8766 name={\glxtrlongshortdescname},
8767 sort={\glxtrlongshortdescsort},%
8768 first={\protect\glsfirstlongemfont{\the\glslongtok}%
8769 \protect\glxtrfullsep{\the\glslabeltok}%
8770 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
8771 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
8772 \protect\glxtrfullsep{\the\glslabeltok}%
8773 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
8774 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
8775 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
8776 }%

```

Unset the regular attribute if it has been set.

```

8777 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8778 \glshasattribute{\the\glslabeltok}{regular}%
8779 {%
8780 \glssetattribute{\the\glslabeltok}{regular}{false}%
8781 }%
8782 {}%
8783 }%
8784 }%
8785 {%
8786 \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
8787 }

```

short-em-long Now the short (long) version

```

8788 \newabbreviationstyle{short-em-long}%
8789 {%
8790 \renewcommand*{\CustomAbbreviationFields}{%
8791 name={\glxtrshortlongname},

```

```

8792 sort={\the\glsshorttok},
8793 description={\the\glslongtok},%
8794 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
8795 \protect\glsxtrfullsep{\the\glslabeltok}%
8796 \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8797 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
8798 \protect\glsxtrfullsep{\the\glslabeltok}%
8799 \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8800 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8801 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8802 \glsattribute{\the\glslabeltok}{regular}%
8803 {%
8804 \glssetattribute{\the\glslabeltok}{regular}{false}%
8805 }%
8806 {}%
8807 }%
8808 }%
8809 {%

```

Mostly as short-long style:

```

8810 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsuffix}%
8811 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
8812 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
8813 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8814 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8815 \renewcommand*{\glsxtrfullformat}[2]{%
8816 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8817 \ifglxtrinsertinside\else##2\fi
8818 \glsxtrfullsep{##1}%
8819 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8820 }%
8821 \renewcommand*{\glsxtrfullplformat}[2]{%
8822 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8823 \ifglxtrinsertinside\else##2\fi
8824 \glsxtrfullsep{##1}%
8825 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8826 }%
8827 \renewcommand*{\Glsxtrfullformat}[2]{%
8828 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8829 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8830 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8831 }%
8832 \renewcommand*{\Glsxtrfullplformat}[2]{%
8833 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8834 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8835 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8836 }%

```

8837 }

rt-em-long-desc As before but user provides description

```
8838 \newabbreviationstyle{short-em-long-desc}%
8839 {%
8840   \renewcommand*{\CustomAbbreviationFields}{%
8841     name={\glstrshortlongdescname},
8842     sort={\glstrshortlongdescsort},
8843     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
8844       \protect\glstrfullsep{\the\glslabeltok}%
8845       \glstrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8846     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
8847       \protect\glstrfullsep{\the\glslabeltok}%
8848       \glstrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8849     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
8850     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
8851   }%
```

Unset the regular attribute if it has been set.

```
8852   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8853     \glshasattribute{\the\glslabeltok}{regular}%
8854     {%
8855       \glissetattribute{\the\glslabeltok}{regular}{false}%
8856     }%
8857   {}%
8858 }%
8859 }%
8860 {%
8861   \GlsXtrUseAbbrStyleFmts{short-em-long}%
8862 }
```

hort-em-long-em

```
8863 \newabbreviationstyle{short-em-long-em}%
8864 {%
8865   \renewcommand*{\CustomAbbreviationFields}{%
8866     name={\glstrshortlongname},
8867     sort={\the\glsshorttok},
8868     description={\protect\glslongemfont{\the\glslongtok}},%
8869     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
8870       \protect\glstrfullsep{\the\glslabeltok}%
8871       \glstrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
8872     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
8873       \protect\glstrfullsep{\the\glslabeltok}%
8874       \glstrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
8875     plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
8876   }%
```

Unset the regular attribute if it has been set.

```

8876 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8877   \glshasattribute{\the\glslabeltok}{regular}}%
8878   {%
8879     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
8880   }%
8881   {}%
8882 }%
8883 }%
8884 {%
8885 \renewcommand*{\abbrvpluralsuffix}{\protect\glstxtremsuffix}%
8886 \renewcommand*{\glssabbrvfnt}[1]{\glssabbrvemfont{##1}}%
8887 \renewcommand*{\glssfirstabbrvfnt}[1]{\glssfirstabbrvemfont{##1}}%
8888 \renewcommand*{\glssfirstlongfnt}[1]{\glssfirstlongemfont{##1}}%
8889 \renewcommand*{\glsslongfnt}[1]{\glsslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

8890 \renewcommand*{\glssxtrfullformat}[2]{%
8891   \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
8892   \ifglssxtrininsertinside\else##2\fi
8893   \glssxtrfullsep{##1}%
8894   \glssxtrparen{\glssfirstlongemfont{\glssaccesslong{##1}}}%
8895 }%
8896 \renewcommand*{\glssxtrfullplformat}[2]{%
8897   \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
8898   \ifglssxtrininsertinside\else##2\fi
8899   \glssxtrfullsep{##1}%
8900   \glssxtrparen{\glssfirstlongemfont{\glssaccesslongpl{##1}}}%
8901 }%
8902 \renewcommand*{\Glsxtrfullformat}[2]{%
8903   \glssfirstabbrvemfont{\Glsaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
8904   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
8905   \glssxtrparen{\glssfirstlongemfont{\glssaccesslong{##1}}}%
8906 }%
8907 \renewcommand*{\Glsxtrfullplformat}[2]{%
8908   \glssfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
8909   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
8910   \glssxtrparen{\glssfirstlongemfont{\glssaccesslongpl{##1}}}%
8911 }%
8912 }

```

em-long-em-desc

```

8913 \newabbreviationstyle{short-em-long-em-desc}%
8914 {%
8915 \renewcommand*{\CustomAbbreviationFields}{%
8916   name={\glssxtrshortlongdescname},%
8917   sort={\glssxtrshortlongdescsort},%
8918   first={\protect\glssfirstabbrvemfont{\the\glssshorttok}}%
8919   \protect\glssxtrfullsep{\the\glslabeltok}}%
8920   \glssxtrparen{\protect\glssfirstlongemfont{\the\glsslongtok}}},%
8921   firstplural={\protect\glssfirstabbrvemfont{\the\glssshortpltok}}%

```

```

8922 \protect\glxtrfullsep{\the\glslabeltok}%
8923 \glxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
8924 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
8925 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
8926 }%

```

Unset the regular attribute if it has been set.

```

8927 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8928 \glshasattribute{\the\glslabeltok}{regular}%
8929 {%
8930 \glissetattribute{\the\glslabeltok}{regular}{false}%
8931 }%
8932 {}%
8933 }%
8934 }%
8935 {%
8936 \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
8937 }

```

short-em

```

8938 \newabbreviationstyle{short-em}%
8939 {%
8940 \renewcommand*{\CustomAbbreviationFields}{%
8941 name={\glxtrshortnolongname},
8942 sort={\the\glsshorttok},
8943 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
8944 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
8945 text={\protect\glsabbrvemfont{\the\glsshorttok}},
8946 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
8947 description={\the\glslongtok}}%
8948 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8949 \glissetattribute{\the\glslabeltok}{regular}{true}}%
8950 }%
8951 {%
8952 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
8953 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
8954 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
8955 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8956 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8957 \renewcommand*{\glxtrinlinefullformat}[2]{%
8958 \protect\glsfirstabbrvemfont{\glsaccessshort{##1}}%
8959 \ifglxtrininsertinside##2\fi}%
8960 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
8961 \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8962 }%
8963 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8964 \protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}%
8965 \ifglxtrininsertinside##2\fi}%

```

```

8966 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8967 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8968 }%

8969 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8970 \protect\glsfirstabbrvemfont{\Glsaccessshort{##1}%
8971 \ifglxtrinsertinside##2\fi}%
8972 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8973 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
8974 }%

8975 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8976 \protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}%
8977 \ifglxtrinsertinside##2\fi}%
8978 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8979 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
8980 }%

```

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.

```

8981 \renewcommand*{\glxtrfullformat}[2]{%
8982 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8983 \ifglxtrinsertinside\else##2\fi
8984 }%
8985 \renewcommand*{\glxtrfullplformat}[2]{%
8986 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8987 \ifglxtrinsertinside\else##2\fi
8988 }%
8989 \renewcommand*{\Glsxtrfullformat}[2]{%
8990 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8991 \ifglxtrinsertinside\else##2\fi
8992 }%
8993 \renewcommand*{\Glsxtrfullplformat}[2]{%
8994 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8995 \ifglxtrinsertinside\else##2\fi
8996 }%
8997 }

```

short-em-nolong

```
8998 \letabbreviationstyle{short-em-nolong}{short-em}
```

short-em-desc

```

8999 \newabbreviationstyle{short-em-desc}%
9000 {%
9001 \renewcommand*{\CustomAbbreviationFields}{%
9002 name={\glxtrshortdescname},
9003 sort={\the\glssshorttok},
9004 first={\protect\glsfirstabbrvemfont{\the\glssshorttok}},
9005 firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}},
9006 text={\protect\glssabbrvemfont{\the\glssshorttok}},

```

```

9007 plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
9008 description={\the\glslongtok}}%
9009 \renewcommand*\GlsXtrPostNewAbbreviation{%
9010 \glssetattribute{\the\glslabeltok}{regular}{true}}%
9011 }%
9012 {%
9013 \renewcommand*\abbrvpluralsuffix{\protect\glsxtremsuffix}%
9014 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
9015 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
9016 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9017 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9018 \renewcommand*\glsxtrinlinefullformat[2]{%
9019 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9020 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9021 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9022 }%
9023 \renewcommand*\glsxtrinlinefullplformat[2]{%
9024 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9025 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9026 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9027 }%
9028 \renewcommand*\Glsxtrinlinefullformat[2]{%
9029 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9030 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9031 \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
9032 }%
9033 \renewcommand*\Glsxtrinlinefullplformat[2]{%
9034 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9035 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9036 \glsxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
9037 }%

```

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.

```

9038 \renewcommand*\glsxtrfullformat[2]{%
9039 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9040 \ifglsxtrininsertinside\else##2\fi
9041 }%
9042 \renewcommand*\glsxtrfullplformat[2]{%
9043 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9044 \ifglsxtrininsertinside\else##2\fi
9045 }%
9046 \renewcommand*\Glsxtrfullformat[2]{%
9047 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9048 \ifglsxtrininsertinside\else##2\fi
9049 }%
9050 \renewcommand*\Glsxtrfullplformat[2]{%
9051 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%

```

```

9052     \ifglxtrinsertinside\else##2\fi
9053 }%
9054 }

```

-em-nolong-desc

```

9055 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

9056 \newabbreviationstyle{nolong-short-em}%
9057 {%
9058   \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
9059 }%
9060 {%
9061   \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9062   \renewcommand*{\glxtrinlinefullformat}[2]{%
9063     \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9064       \ifglxtrinsertinside##2\fi}%
9065     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9066     \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9067   }%
9068   \renewcommand*{\glxtrinlinefullplformat}[2]{%
9069     \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9070       \ifglxtrinsertinside##2\fi}%
9071     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9072     \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9073   }%
9074   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9075     \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9076       \ifglxtrinsertinside##2\fi}%
9077     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9078     \glxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9079   }%
9080   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9081     \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9082       \ifglxtrinsertinside##2\fi}%
9083     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9084     \glxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9085   }%
9086 }

```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9087 \newabbreviationstyle{long-noshort-em}%
9088 {%
9089   \renewcommand*{\CustomAbbreviationFields}{%
9090     name={\glxtrlongnoshortname},
9091     sort={\the\glsshorttok},
9092     first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},

```

```

9093   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9094   text={\protect\glslongdefaultfont{\the\glslongtok}},
9095   plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9096   description={\the\glslongtok}%
9097 }%
9098 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9099   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9100 }%
9101 {%
9102   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
9103   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9104   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9105   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9106   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9107   \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9108     \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9109     \ifglsxtrininsertinside \else##2\fi
9110   }%
9111   \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9112     \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9113     \ifglsxtrininsertinside \else##2\fi
9114   }%
9115   \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9116     \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9117     \ifglsxtrininsertinside \else##2\fi
9118   }%
9119   \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9120     \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9121     \ifglsxtrininsertinside \else##2\fi
9122   }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9123   \renewcommand*{\glsxtrinlinefullformat}[2]{%
9124     \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9125     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9126     \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9127   }%
9128   \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9129     \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9130     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9131     \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9132   }%
9133   \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9134     \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9135     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9136     \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9137   }%
9138   \renewcommand*{\Glsxtrinlinefullplformat}[2]{%

```

```

9139 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9140 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9141 \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9142 }%

```

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.

```

9143 \renewcommand*{\glxtrfullformat}[2]{%
9144 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9145 \ifglxtrinsertinside\else##2\fi
9146 }%
9147 \renewcommand*{\glxtrfullplformat}[2]{%
9148 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9149 \ifglxtrinsertinside\else##2\fi
9150 }%
9151 \renewcommand*{\Glsxtrfullformat}[2]{%
9152 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9153 \ifglxtrinsertinside\else##2\fi
9154 }%
9155 \renewcommand*{\Glsxtrfullplformat}[2]{%
9156 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9157 \ifglxtrinsertinside\else##2\fi
9158 }%
9159 }

```

long-em Backward compatibility:

```

9160 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

9161 \newabbreviationstyle{long-em-noshort-em}%
9162 {%
9163 \renewcommand*{\CustomAbbreviationFields}{%
9164 name={\glxtrlongnoshortname},
9165 sort={\the\glsshorttok},
9166 first={\protect\glsfirstlongemfont{\the\glslongtok}},
9167 firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9168 text={\protect\glslongemfont{\the\glslongtok}},
9169 plural={\protect\glslongemfont{\the\glslongpltok}},%
9170 description={\protect\glslongemfont{\the\glslongtok}}%
9171 }%
9172 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9173 \glsssetattribute{\the\glslabeltok}{regular}{true}}%
9174 }%
9175 {%
9176 \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9177 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9178 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9179 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
9180 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9181 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9182   \glslongemfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9183   \ifglxtrinsertinside \else##2\fi
9184 }%
9185 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9186   \glslongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9187   \ifglxtrinsertinside \else##2\fi
9188 }%
9189 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9190   \glslongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9191   \ifglxtrinsertinside \else##2\fi
9192 }%
9193 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9194   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9195   \ifglxtrinsertinside \else##2\fi
9196 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9197 \renewcommand*{\glxtrinlinefullformat}[2]{%
9198   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9199   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9200   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9201 }%
9202 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9203   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9204   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9205   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9206 }%
9207 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9208   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9209   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9210   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9211 }%
9212 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9213   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9214   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9215   \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9216 }%

```

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.

```

9217 \renewcommand*{\glxtrfullformat}[2]{%
9218   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9219   \ifglxtrinsertinside\else##2\fi
9220 }%
9221 \renewcommand*{\glxtrfullplformat}[2]{%
9222   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9223   \ifglxtrinsertinside\else##2\fi
9224 }%

```

```

9225 \renewcommand*{\Glsxtrfullformat}[2]{%
9226   \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9227   \ifglxtrinsertinside\else##2\fi
9228 }%
9229 \renewcommand*{\Glsxtrfullplformat}[2]{%
9230   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9231   \ifglxtrinsertinside\else##2\fi
9232 }%
9233 }

```

`noshort-em-noreg` Like `long-em-noshort-em` but doesn't set the regular attribute.

```

9234 \newabbreviationstyle{long-em-noshort-em-noreg}%
9235 {%
9236   \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
9237   \GlsXtrPostNewAbbreviation{%
9238     \glshasattribute{\the\glslabeltok}{regular}%
9239     {%
9240       \glissetattribute{\the\glslabeltok}{regular}{false}%
9241     }%
9242   }%
9243 }%
9244 }%
9245 {%
9246   \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
9247 }

```

`noshort-em-desc` The emphasized font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9248 \newabbreviationstyle{long-noshort-em-desc}%
9249 {%
9250   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9251 }%
9252 {%
9253   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9254   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9255   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9256   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9257   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
9258   \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9259     \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9260     \ifglxtrinsertinside \else##2\fi
9261   }%
9262   \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9263     \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9264     \ifglxtrinsertinside \else##2\fi
9265   }%

```

The format for subsequent use (not used when the regular attribute is set).

```

9266 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9267   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9268   \ifglxtrinsertinside \else##2\fi
9269 }%
9270 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9271   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9272   \ifglxtrinsertinside \else##2\fi
9273 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9274 \renewcommand*{\glxtrinlinefullformat}[2]{%
9275   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9276   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9277   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9278 }%
9279 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9280   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9281   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9282   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9283 }%
9284 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9285   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9286   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9287   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
9288 }%
9289 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9290   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9291   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9292   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
9293 }%

```

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.

```

9294 \renewcommand*{\glxtrfullformat}[2]{%
9295   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9296   \ifglxtrinsertinside\else##2\fi
9297 }%
9298 \renewcommand*{\glxtrfullplformat}[2]{%
9299   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9300   \ifglxtrinsertinside\else##2\fi
9301 }%
9302 \renewcommand*{\Glsxtrfullformat}[2]{%
9303   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9304   \ifglxtrinsertinside\else##2\fi
9305 }%
9306 \renewcommand*{\Glsxtrfullplformat}[2]{%
9307   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9308   \ifglxtrinsertinside\else##2\fi
9309 }%
9310 }

```

long-desc-em Backward compatibility:

```
9311 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

noshort-em-desc The short form is explicitly invoked through commands like `\glsshort`. The long form is emphasized.

```
9312 \newabbreviationstyle{long-em-noshort-em-desc}%
9313 {%
9314   \renewcommand*{\CustomAbbreviationFields}{%
9315     name={\glxtrlongnoshortdescname},
9316     sort={\the\glslongtok},
9317     first={\protect\glsfirstlongemfont{\the\glslongtok}},
9318     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
9319     text={\glslongemfont{\the\glslongtok}},
9320     plural={\glslongemfont{\the\glslongpltok}}}%
9321   }%
9322   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9323     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9324 }%
9325 {%
9326   \renewcommand*{\abbrvpluralsuffix}{\protect\glxxtremsuffix}%
9327   \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
9328   \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
9329   \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongemfont{##1}}%
9330   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
9331 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9332   \glslongemfont{\glssaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9333   \ifglxtrininsertinside \else##2\fi
9334 }%
9335 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9336   \glslongemfont{\glssaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9337   \ifglxtrininsertinside \else##2\fi
9338 }%
9339 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9340   \glslongemfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
9341   \ifglxtrininsertinside \else##2\fi
9342 }%
9343 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9344   \glslongemfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
9345   \ifglxtrininsertinside \else##2\fi
9346 }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
9347 \renewcommand*{\glxtrininlinefullformat}[2]{%
9348   \glsfirstlongemfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9349   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9350   \glxtrparen{\protect\glssfirstabbrvemfont{\glssaccessshort{##1}}}%
9351 }%
9352 \renewcommand*{\glxtrininlinefullplformat}[2]{%
```

```

9353 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9354 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9355 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9356 }%
9357 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9358 \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9359 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9360 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
9361 }%
9362 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9363 \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9364 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9365 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
9366 }%

```

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.

```

9367 \renewcommand*{\glxtrfullformat}[2]{%
9368 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9369 \ifglxtrinsertinside\else##2\fi
9370 }%
9371 \renewcommand*{\glxtrfullplformat}[2]{%
9372 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9373 \ifglxtrinsertinside\else##2\fi
9374 }%
9375 \renewcommand*{\Glsxtrfullformat}[2]{%
9376 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9377 \ifglxtrinsertinside\else##2\fi
9378 }%
9379 \renewcommand*{\Glsxtrfullplformat}[2]{%
9380 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9381 \ifglxtrinsertinside\else##2\fi
9382 }%
9383 }

```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```

9384 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
9385 {%
9386 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%

```

Unset the regular attribute if it has been set.

```

9387 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9388 \glshasattribute{\the\glslabeltok}{regular}%
9389 {%
9390 \glissetattribute{\the\glslabeltok}{regular}{false}%
9391 }%
9392 {}%
9393 }%
9394 }%
9395 {%

```

```

9396 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
9397 }

```

ort-em-footnote

```

9398 \newabbreviationstyle{short-em-footnote}%
9399 {%
9400 \renewcommand*{\CustomAbbreviationFields}{%
9401   name={\glsxtrfootnotename},
9402   sort={\the\glsshorttok},
9403   description={\the\glslongtok},%
9404   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
9405     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9406     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9407   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
9408     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9409     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9410   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.

```

9411 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9412 \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9413 \glshasattribute{\the\glslabeltok}{regular}%
9414 {%
9415 \glsssetattribute{\the\glslabeltok}{regular}{false}%
9416 }%
9417 {}%
9418 }%
9419 }%
9420 {%
9421 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9422 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
9423 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
9424 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9425 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9426 \renewcommand*{\glxtrfullformat}[2]{%
9427 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9428 \ifglxtrininsertinside\else##2\fi
9429 \protect\glxtrabbrvfootnote{##1}%
9430 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9431 }%
9432 \renewcommand*{\glxtrfullplformat}[2]{%
9433 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9434 \ifglxtrininsertinside\else##2\fi
9435 \protect\glxtrabbrvfootnote{##1}%
9436 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9437 }%
9438 \renewcommand*{\Glsxtrfullformat}[2]{%

```

```

9439 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9440 \ifglxtrinsertinside\else##2\fi
9441 \protect\glxtrabbrvfootnote{##1}%
9442 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9443 }%
9444 \renewcommand*{\Glsxtrfullplformat}[2]{%
9445 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9446 \ifglxtrinsertinside\else##2\fi
9447 \protect\glxtrabbrvfootnote{##1}%
9448 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9449 }%

```

The first use full form and the inline full form use the short (long) style.

```

9450 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9451 \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9452 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9453 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9454 }%
9455 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9456 \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9457 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9458 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9459 }%
9460 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9461 \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9462 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9463 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9464 }%
9465 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9466 \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9467 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9468 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9469 }%
9470 }

```

footnote-em Backward compatibility:

```

9471 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}

```

em-postfootnote

```

9472 \newabbreviationstyle{short-em-postfootnote}%
9473 {%
9474 \renewcommand*{\CustomAbbreviationFields}{%
9475 name={\glxtrfootnotename},
9476 sort={\the\glsshorttok},
9477 description={\the\glslongtok},%
9478 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
9479 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
9480 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.

```
9481 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9482   \csdef{glxstrpostlink\glscategorylabel}{%
9483     \glxtrifwasfirstuse
9484     {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
9485       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9486       {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
9487   }%
9488   {%
9489   }%
9490   \glshasattribute{\the\glslabelltok}{regular}%
9491   {%
9492     \glissetattribute{\the\glslabelltok}{regular}{false}%
9493   }%
9494   {%
9495   }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
9496 \renewcommand*{\glxtrsetupfulldefs}{%
9497   \let\glxtrifwasfirstuse\@secondoftwo
9498 }%
9499}%
9500{%
9501 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtremsuffix}%
9502 \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
9503 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
9504 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongfootnotefont{##1}}%
9505 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
9506 \renewcommand*{\glxtrfullformat}[2]{%
9507   \glfirstabbrvemfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9508   \ifglxtrininsertinside\else##2\fi
9509 }%
9510 \renewcommand*{\glxtrfullplformat}[2]{%
9511   \glfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9512   \ifglxtrininsertinside\else##2\fi
9513 }%
9514 \renewcommand*{\Glsxtrfullformat}[2]{%
9515   \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9516   \ifglxtrininsertinside\else##2\fi
9517 }%
9518 \renewcommand*{\Glsxtrfullplformat}[2]{%
9519   \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9520   \ifglxtrininsertinside\else##2\fi
```

```

9521 }%
    The first use full form and the inline full form use the short (long) style.
9522 \renewcommand*{\glxstrinlinefullformat}[2]{%
9523   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglxstrinsertinside##2\fi}%
9524   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9525   \glxstrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9526 }%
9527 \renewcommand*{\glxstrinlinefullplformat}[2]{%
9528   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglxstrinsertinside##2\fi}%
9529   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9530   \glxstrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9531 }%
9532 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9533   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxstrinsertinside##2\fi}%
9534   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9535   \glxstrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9536 }%
9537 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9538   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxstrinsertinside##2\fi}%
9539   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
9540   \glxstrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9541 }%
9542 }

```

postfootnote-em Backward compatibility:

```

9543 \@glxstr@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:

glxstruserfield Default is the useri field.

```

9544 \newcommand*{\glxstruserfield}{useri}

```

glxstruserparen 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.

```

9545 \ifdef\glscurrentfieldvalue
9546 {
9547   \newcommand*{\glxstruserparen}[2]{%
9548     \glxstrfullsep{#2}%
9549     \glxstrparen
9550     {#1\ifglshasfield{\glxstruserfield}{#2}{, \glscurrentfieldvalue}}}%
9551   }
9552 }
9553 {
9554   \newcommand*{\glxstruserparen}[2]{%

```

```

9555 \glxtrfullsep{#2}%
9556 \glxtrparen
9557   {#1\ifglshasfield{\glxtruserfield}{#2}{, \@glo@thisvalue}{}}%
9558 }
9559 }

```

Font used for short form:

lsabbrvuserfont

```
9560 \newcommand*{\glsabbrvuserfont}[1]{\glsabbrvdefaultfont{#1}}
```

Font used for short form on first use:

stabbrvuserfont

```
9561 \newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
9562 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
9563 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
9564 \newcommand*{\glsxtrusersuffix}{\glxtrabbrvpluralsuffix}
```

long-short-user

```

9565 \newabbreviationstyle{long-short-user}%
9566 {%
9567   \renewcommand*{\CustomAbbreviationFields}{%
9568     name={\glxtrlongshortname},
9569     sort={\the\glsshorttok},
9570     first={\protect\glsfirstlonguserfont{\the\glslongtok}}%
9571     \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
9572     {\the\glslabeltok}},%
9573     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}}%
9574     \protect\glxtruserparen
9575     {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%

9576     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9577     description={\protect\glslonguserfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

9578 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9579   \glshasattribute{\the\glslabeltok}{regular}%
9580   {%

```

```

9581 \glsssetAttribute{\the\glslabelltok}{regular}{false}%
9582 }%
9583 {}%
9584 }%
9585 }%
9586 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

9587 \renewcommand*\abbrvpluralsuffix{\glsxtrusersuffix}%
9588 \renewcommand*\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
9589 \renewcommand*\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
9590 \renewcommand*\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
9591 \renewcommand*\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9592 \renewcommand*\glsxtrfullformat}[2]{%
9593 \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9594 \ifglxtrinsertinside\else##2\fi
9595 \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
9596 }%
9597 \renewcommand*\glsxtrfullplformat}[2]{%
9598 \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9599 \ifglxtrinsertinside\else##2\fi
9600 \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9601 }%
9602 \renewcommand*\Glsxtrfullformat}[2]{%
9603 \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9604 \ifglxtrinsertinside\else##2\fi
9605 \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
9606 }%
9607 \renewcommand*\Glsxtrfullplformat}[2]{%
9608 \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9609 \ifglxtrinsertinside\else##2\fi
9610 \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9611 }%
9612 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

9613 \newabbreviationstyle{long-postshort-user}%
9614 {%
9615 \renewcommand*\CustomAbbreviationFields{%
9616 name={\glsxtrlongshortname},
9617 sort={\the\glsshorttok},
9618 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
9619 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
9620 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9621 description={\protect\glslonguserfont{\the\glslongtok}}}%
9622 \renewcommand*\GlsXtrPostNewAbbreviation{%
9623 \csdef{glsxtrpostlink\glscategorylabel}{%

```

```

9624     \glxtrifwasfirstuse
9625     {%
9626         \glxtruserparen
9627         {\glsfirstabbrvuserfont{\gl Sentryshort{\glslabel}}}%
9628         {\glslabel}%
9629     }%
9630     {}%
9631 }%
9632 \glshasattribute{\the\glslabeltok}{regular}%
9633 {%
9634     \glissetattribute{\the\glslabeltok}{regular}{false}%
9635 }%
9636 {}%
9637 }%
9638 }%
9639 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

9640 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
9641 \renewcommand*{\glabbrvfont}[1]{\glabbrvuserfont{##1}}%
9642 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
9643 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
9644 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

9645 \renewcommand*{\glxtrfullformat}[2]{%
9646     \glsfirstlonguserfont{\glaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9647     \ifglxtrininsertinside\else##2\fi
9648 }%
9649 \renewcommand*{\glxtrfullplformat}[2]{%
9650     \glsfirstlonguserfont{\glaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9651     \ifglxtrininsertinside\else##2\fi
9652 }%
9653 \renewcommand*{\Glsxtrfullformat}[2]{%
9654     \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9655     \ifglxtrininsertinside\else##2\fi
9656 }%
9657 \renewcommand*{\Glsxtrfullplformat}[2]{%
9658     \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9659     \ifglxtrininsertinside\else##2\fi
9660 }%

```

In-line format:

```

9661 \renewcommand*{\glxtrininlinefullformat}[2]{%
9662     \glsfirstlonguserfont{\glaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9663     \ifglxtrininsertinside\else##2\fi
9664     \glxtruserparen{\glsfirstabbrvuserfont{\glaccessshort{##1}}}{##1}%
9665 }%
9666 \renewcommand*{\glxtrininlinefullplformat}[2]{%
9667     \glsfirstlonguserfont{\glaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%

```

```

9668 \ifglxtrinsertinside\else##2\fi
9669 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9670 }%
9671 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9672 \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9673 \ifglxtrinsertinside\else##2\fi
9674 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
9675 }%
9676 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9677 \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9678 \ifglxtrinsertinside\else##2\fi
9679 \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
9680 }%
9681 }

```

ortuserdescname

```

9682 \newcommand*{\glxtrlongshortuserdescname}{%
9683 \protect\glslonguserfont{\the\glslongtok}%
9684 \protect\glxtruserparen
9685 {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
9686 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

9687 \newabbreviationstyle{long-postshort-user-desc}{%
9688 {%
9689 \renewcommand*{\CustomAbbreviationFields}{%
9690 name={\glxtrlongshortuserdescname},
9691 sort={\the\glslongtok},
9692 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
9693 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
9694 text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
9695 plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
9696 }%
9697 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9698 \csdef{glxtrpostlink\glscategorylabel}{%
9699 \glxtrifwasfirstuse
9700 {%
9701 \glxtruserparen
9702 {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
9703 {\glslabel}%
9704 }%
9705 }%
9706 }%
9707 \glshasattribute{\the\glslabeltok}{regular}%
9708 {%
9709 \glssetattribute{\the\glslabeltok}{regular}{false}%
9710 }%
9711 }%

```

```

9712 }%
9713 }%
9714 {%
9715   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
9716 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

9717 \newabbreviationstyle{short-postlong-user}%
9718 {%
9719   \renewcommand*{\CustomAbbreviationFields}{%
9720     name={\glstrshortlongname},
9721     sort={\the\glsshorttok},
9722     first={\protect\glfirstlonguserfont{\the\glslongtok}},%
9723     firstplural={\protect\glfirstlonguserfont{\the\glslongpltok}},%
9724     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
9725     description={\protect\glslonguserfont{\the\glslongtok}}}%
9726 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9727   \csdef{glstrpostlink\glscategorylabel}{%
9728     \glstrifwasfirstuse
9729     {%
9730       \glstruserparen
9731       {\glfirstlonguserfont{\glstentrylong{\glslabel}}}%
9732       {\glslabel}%
9733     }%
9734     }%
9735   }%
9736   \glshasattribute{\the\glslabeltok}{regular}%
9737   {%
9738     \glissetattribute{\the\glslabeltok}{regular}{false}%
9739   }%
9740   }%
9741 }%
9742 }%
9743 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

9744 \renewcommand*{\abbrvpluralsuffix}{\glstrusersuffix}%
9745 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
9746 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
9747 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
9748 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

9749 \renewcommand*{\glstrfullformat}[2]{%
9750   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9751   \ifglstrinsertinside\else##2\fi
9752 }%
9753 \renewcommand*{\glstrfullplformat}[2]{%
9754   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%

```

```

9755 \ifglxtrinsertinside\else##2\fi
9756 }%
9757 \renewcommand*{\Glsxtrfullformat}[2]{%
9758 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9759 \ifglxtrinsertinside\else##2\fi
9760 }%
9761 \renewcommand*{\Glsxtrfullplformat}[2]{%
9762 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9763 \ifglxtrinsertinside\else##2\fi
9764 }%

```

In-line format:

```

9765 \renewcommand*{\glxtrinlinefullformat}[2]{%
9766 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9767 \ifglxtrinsertinside\else##2\fi
9768 \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
9769 }%
9770 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9771 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9772 \ifglxtrinsertinside\else##2\fi
9773 \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
9774 }%
9775 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9776 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9777 \ifglxtrinsertinside\else##2\fi
9778 \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
9779 }%
9780 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9781 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9782 \ifglxtrinsertinside\else##2\fi
9783 \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
9784 }%
9785 }

```

onguserdesname

```

9786 \newcommand*{\glxtrshortlonguserdesname}{%
9787 \protect\glssabbrvuserfont{\the\glssshorttok}%
9788 \protect\glxtruserparen
9789 {\protect\glslonguserfont{\the\glslongpltok}}%
9790 {\the\glslabelltok}%
9791 }

```

long-user-desc Like short-postlong-user but leaves the user to specify the description.

```

9792 \newabbreviationstyle{short-postlong-user-desc}%
9793 {%
9794 \renewcommand*{\CustomAbbreviationFields}{%
9795 name={\glxtrshortlonguserdesname},
9796 sort={\the\glssshorttok},
9797 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
9798 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

```

```

9799     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
9800     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
9801 }%
9802 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9803   \csdef{glsxtrpostlink\glscategorylabel}{%
9804     \glsxtrifwasfirstuse
9805     {%
9806       \glsxtruserparen
9807       {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
9808       {\glslabel}}%
9809     }%
9810   }%
9811 }%
9812 \glsattribute{\the\glslabeltok}{regular}%
9813 {%
9814   \glssetattribute{\the\glslabeltok}{regular}{false}%
9815 }%
9816 {}%
9817 }%
9818 }%
9819 {%
9820   \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
9821 }

```

short-user-desc

```

9822 \newabbreviationstyle{long-short-user-desc}%
9823 {%
9824   \renewcommand*{\CustomAbbreviationFields}{%
9825     name={\glsxtrlongshortuserdescname},
9826     sort={\glsxtrlongshortdescsort},%
9827     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
9828       \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
9829       {\the\glslabeltok}},%
9830     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
9831       \protect\glsxtruserparen
9832       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
9833     text={\protect\glsabbrvfont{\the\glsshorttok}},%
9834     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
9835 }%

```

Unset the regular attribute if it has been set.

```

9836 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9837   \glsattribute{\the\glslabeltok}{regular}%
9838   {%
9839     \glssetattribute{\the\glslabeltok}{regular}{false}%
9840   }%
9841   {}%
9842 }%
9843 }%

```

```

9844 {%
9845   \GlsXtrUseAbbrStyleFmts{long-short-user}%
9846 }

```

short-long-user

```

9847 \newabbreviationstyle{short-long-user}%
9848 {%
    \glslonguserfont is used in the description since \glsdesc doesn't set the style.
9849   \renewcommand*{\CustomAbbreviationFields}{%
9850     name={\glstrshortlongname},
9851     sort={\the\glsshorttok},
9852     description={\protect\glslonguserfont{\the\glslongtok}},%
9853     first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
9854       \protect\glstruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
9855     {\the\glslabeltok}},%
9856     firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
9857       \protect\glstruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
9858     {\the\glslabeltok}},%
9859     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

9860 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9861   \glshasattribute{\the\glslabeltok}{regular}%
9862   {%
9863     \glssetattribute{\the\glslabeltok}{regular}{false}%
9864   }%
9865   {}%
9866 }%
9867 }%
9868 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

9869 \renewcommand*{\abbrvpluralsuffix}{\glstrusersuffix}%
9870 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
9871 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
9872 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
9873 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

9874 \renewcommand*{\glstrfullformat}[2]{%
9875   \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9876   \ifglstrinsertinside\else##2\fi
9877   \glstruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
9878 }%
9879 \renewcommand*{\glstrfullplformat}[2]{%
9880   \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9881   \ifglstrinsertinside\else##2\fi
9882   \glstruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
9883 }%

```

```

9884 \renewcommand*{\Glsxtrfullformat}[2]{%
9885   \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9886   \ifglxtrinsertinside\else##2\fi
9887   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
9888 }%
9889 \renewcommand*{\Glsxtrfullplformat}[2]{%
9890   \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9891   \ifglxtrinsertinside\else##2\fi
9892   \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
9893 }%
9894 }

```

-long-user-desc

```

9895 \newabbreviationstyle{short-long-user-desc}%
9896 {%
9897   \renewcommand*{\CustomAbbreviationFields}{%
9898     name={\glxtrshortlonguserdescname},
9899     sort={\glxtrshortlongdescsort},%

9900     first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
9901       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}}%
9902     {\the\glslabeltok}},%
9903     firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
9904       \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}}%
9905     {\the\glslabeltok}},%
9906     text={\protect\glsabbrvfont{\the\glsshorttok}},%
9907     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
9908   }%

   Unset the regular attribute if it has been set.
9909   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9910     \glshasattribute{\the\glslabeltok}{regular}%
9911     {%
9912       \glissetattribute{\the\glslabeltok}{regular}{false}%
9913     }%
9914   }%
9915 }%
9916 }%
9917 {%
9918   \GlsXtrUseAbbrStyleFmts{short-long-user}%
9919 }

```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the `markwords` attribute. They check if the inserted material (provided by the final optional argument of commands like `\gls`) starts with a hyphen. If it does, the insert is added to the parenthetical material. Note that commands like `\glxtrlong set \glsinsert` to empty with the entire link-text stored in `\glscustomtext`.

trifhyphenstart Checks if the argument starts with a hyphen. The argument may be `\glsinsert` so check for that and expand.

```
9920 \newrobustcmd*{\glstrifhyphenstart}[3]{%
9921   \ifx\glsinsert#1\relax
9922   \expandafter\glstrifhyphenstart#1\relax\relax
9923   \@end@glstrifhyphenstart{#2}{#3}%
9924   \else
9925   \@glstrifhyphenstart#1\relax\relax\@end@glstrifhyphenstart{#2}{#3}%
9926   \fi
9927 }
```

trifhyphenstart

```
9928 \def\@glstrifhyphenstart#1#2\@end@glstrifhyphenstart#3#4{%
9929   \ifx-#1\relax#3\else #4\fi
9930 }
```

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.

```
9931 \newcommand*{\glstrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
9932 {%
```

If `<insert>` starts with a hyphen, redefine `\glstrwordsep` 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 `\glstrwordsep` if `<insert>` doesn't start with a hyphen.

```
9933   \glstrifhyphenstart{#4}{\def\glstrwordsep{-}}{}%
9934   \glsfirstlonghyphenfont{#2\ifglstrinsertinside{#4}\fi}%
9935   \ifglstrinsertinside\else{#4}\fi
9936   \glstrfullsep{#1}%
9937   \glstrparen{\glsfirstabbrvhyphenfont{#3\ifglstrinsertinside{#4}\fi}%
9938   \ifglstrinsertinside\else{#4}\fi}%
9939 }%
9940 }
```

abbrvhyphenfont

```
9941 \newcommand*{\glsabbrvhyphenfont}{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
9942 \newcommand*{\glsfirstabbrvhyphenfont}{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
9943 \newcommand*{\glslonghyphenfont}{\glslongdefaultfont}%
```

longhyphenfont

```
9944 \newcommand*{\glfirstlonghyphenfont}{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
9945 \newcommand*{\glxtrhyphensuffix}{\glxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the markwords attribute.

```
9946 \newabbreviationstyle{long-hyphen-short-hyphen}%
9947 {%
9948   \renewcommand*{\CustomAbbreviationFields}{%
9949     name={\glxtrlongshortname},
9950     sort={\the\glsshorttok},
9951     first={\protect\glfirstlonghyphenfont{\the\glslongtok}%
9952       \protect\glxtrfullsep{\the\glslabeltok}%
9953       \glxtrparen{\protect\glfirstabbrvhyphenfont{\the\glsshorttok}}},%
9954     firstplural={\protect\glfirstlonghyphenfont{\the\glslongpltok}%
9955       \protect\glxtrfullsep{\the\glslabeltok}%
9956       \glxtrparen{\protect\glfirstabbrvhyphenfont{\the\glsshortpltok}}},%
9957     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}},%
9958     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
```

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   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
9969   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
9970   \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvhyphenfont{##1}}%
9971   \renewcommand*{\glfirstlongfont}[1]{\glfirstlonghyphenfont{##1}}%
9972   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9973   \renewcommand*{\glxtrfullformat}[2]{%
9974     \glxtrlonghyphenshort{##1}{\glssuccesslong{##1}}{\glssuccessshort{##1}}{##2}%
9975   }%
9976   \renewcommand*{\glxtrfullplformat}[2]{%
9977     \glxtrlonghyphenshort{##1}{\glssuccesslongpl{##1}}%
9978     {\glssuccessshortpl{##1}}{##2}%
9979   }%
9980   \renewcommand*{\Glsxtrfullformat}[2]{%
9981     \glxtrlonghyphenshort{##1}{\glssuccesslong{##1}}{\glssuccessshort{##1}}{##2}%
9982   }%
```

```

9983 \renewcommand*{\Glsxtrfullplformat}[2]{%
9984   \glsxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
9985   {\Glsaccessshortpl{##1}}{##2}%
9986 }%
9987 }

```

`ort-hyphen-desc` Like `long-hyphen-short-hyphen` but the description must be supplied by the user.

```

9988 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
9989 {%
9990   \renewcommand*{\CustomAbbreviationFields}{%
9991     name={\glsxtrlongshortdescname},
9992     sort={\glsxtrlongshortdescsort},
9993     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
9994       \protect\glsxtrfullsep{\the\glslabeltok}%
9995       \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
9996     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
9997       \protect\glsxtrfullsep{\the\glslabeltok}%
9998       \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
9999     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10000     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
10001   }%

```

Unset the regular attribute if it has been set.

```

10002 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10003   \glsattribute{\the\glslabeltok}{regular}%
10004   {%
10005     \glssetattribute{\the\glslabeltok}{regular}{false}%
10006   }%
10007   {}}%
10008 }%
10009 }%
10010 {%
10011   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
10012 }

```

`onghyphennoshort` `\glsxtrlonghyphennoshort{<label>}{<long>}{<insert>}`

```

10013 \newcommand*{\glsxtrlonghyphennoshort}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10014 {%

```

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.

```

10015   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
10016   \glsfirstlonghyphenfont{#2\ifglsxtrininsertinside{#3}\fi}%

```

```

10017 \ifglxtrinsertinside\else{#3}\fi
10018 }%
10019 }

```

short-desc-noreg This version doesn't show the short form (except explicitly with \glxtrshort). Since \glxtrshort doesn't support the hyphen switch, the short form just uses the default short-form font command. This style won't work with the regular as the regular form isn't flexible enough.

```

10020 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
10021 {%
10022   \renewcommand*{\CustomAbbreviationFields}{%
10023     name={\glxtrlongnoshortdescname},
10024     sort={\expandonce\glxtrorglong},
10025     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10026     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10027     plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
10028   }%

```

Unset the regular attribute if it has been set.

```

10029 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10030   \glshasattribute{\the\glslabeltok}{regular}%
10031   {%
10032     \glissetattribute{\the\glslabeltok}{regular}{false}%
10033   }%
10034   {}%
10035 }%
10036 }%
10037 {%
10038   \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

10039 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10040 \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
10041 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
10042 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10043 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10044 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10045   \glxtrlonghyphennoshort{##1}{\glssaccesslong{##1}}{##2}%
10046 }%
10047 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10048   \glxtrlonghyphennoshort{##1}{\glssaccesslongpl{##1}}{##2}%
10049 }%
10050 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10051   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10052 }%
10053 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10054   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10055 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10056 \renewcommand*{\glxtrinlinefullformat}[2]{%
10057   \glxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
10058   \glxtrfullsep{##1}%
10059   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
10060 }%
10061 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10062   \glxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
10063   \glxtrfullsep{##1}%
10064   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10065 }%
10066 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10067   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10068   \glxtrfullsep{##1}%
10069   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
10070 }%
10071 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10072   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10073   \glxtrfullsep{##1}%
10074   \glxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
10075 }%

```

The first use full form only displays the long form.

```

10076 \renewcommand*{\glxtrfullformat}[2]{%
10077   \glxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
10078 }%
10079 \renewcommand*{\glxtrfullplformat}[2]{%
10080   \glxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
10081 }%
10082 \renewcommand*{\Glsxtrfullformat}[2]{%
10083   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
10084 }%
10085 \renewcommand*{\Glsxtrfullplformat}[2]{%
10086   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
10087 }%
10088 }

```

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.

```

10089 \newabbreviationstyle{long-hyphen-noshort-noreg}%
10090 {%
10091   \renewcommand*{\CustomAbbreviationFields}{%
10092     name={\glxtrlongnoshortname},
10093     sort={\the\glsshorttok},
10094     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10095     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10096     text={\protect\glslonghyphenfont{\the\glslongtok}},%
10097     plural={\protect\glslonghyphenfont{\the\glslongpltok}},%

```

```

10098     description={\the\glslongtok}%
10099 }%

    Unset the regular attribute if it has been set.
10100 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10101     \glshasattribute{\the\glslabeltok}{regular}%
10102     {%
10103         \glissetattribute{\the\glslabeltok}{regular}{false}%
10104     }%
10105     {}%
10106 }%
10107}%
10108{%
10109     \GlsXtrUseAbbrStyleFmts{long-desc}%
10110}

```

`\glsxtrlonghyphen` `\glsxtrlonghyphen{<long>}{<label>}{<insert>}`

Used by long-hyphen-postshort-hyphen. The `<insert>` is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

10111 \newcommand*{\glsxtrlonghyphen}[3]{%
    Grouping is needed to localise the redefinitions.
10112 {%
10113     \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{%
10114         \glsfirstlonghyphenfont{#1}%
10115     }%
10116 }

```

`\glsxtrposthyphenshort` `\glsxtrposthyphenshort{<label>}{<insert>}`

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like `\glsxtrlonghyphenshort` but omits the `<long>` part. This always uses the singular short form.

```

10117 \newcommand*{\glsxtrposthyphenshort}[2]{%
10118 {%
10119     \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{%
10120         \ifglsxtrininsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
10121         \glsxtrfullsep{#1}%
10122         \glsxtrparen
10123         {\glsfirstabbrvhyphenfont{\glsentryshort{#1}}\ifglsxtrininsertinside{#2}\fi}%
10124         \ifglsxtrininsertinside\else{#2}\fi
10125     }%
10126 }%
10127}

```

hyphensubsequent

`\glxtrposthyphensubsequent{<label>}{<insert>}`

Format in the post-link hook for subsequent use. The label is ignored by default.

```
10128 \newcommand*{\glxtrposthyphensubsequent}[2]{%
10129   \glssabrvfont{\ifglxtrininsertinside {#2}\fi}%
10130   \ifglxtrininsertinside \else{#2}\fi
10131 }
```

postshort-hyphen

Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```
10132 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
10133 {%
10134   \renewcommand*{\CustomAbbreviationFields}{%
10135     name={\glsxtrlongshortname},
10136     sort={\the\glsshorttok},
10137     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10138     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10139     plural={\protect\glssabrvhyphenfont{\the\glsshortpltok}},%
10140     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10141   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10142     \csdef{glxtrpostlink\glscategorylabel}{%
10143       \glsxtrifwasfirstuse
10144       {%
10145         \glxtrposthyphenshort{\glslabel}{\glsinsert}%
10146       }%
10147     }%
```

Put the insertion into the post-link:

```
10148       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10149     }%
10150   }%
10151   \glshasattribute{\the\glslabeltok}{regular}%
10152   {%
10153     \glissetattribute{\the\glslabeltok}{regular}{false}%
10154   }%
10155   {}%
10156 }%
10157 }%
10158 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
10159 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
10160 \renewcommand*{\glssabrvfont}[1]{\glssabrvhyphenfont{##1}}%
10161 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10162 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10163 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

Subsequent use needs to omit the insertion:

```

10164 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10165   \glsabbrvfont{\glsaccessshort{##1}}}%
10166 }%
10167 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10168   \glsabbrvfont{\glsaccessshortpl{##1}}}%
10169 }%
10170 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10171   \glsabbrvfont{\Glsaccessshort{##1}}}%
10172 }%
10173 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10174   \glsabbrvfont{\Glsaccessshortpl{##1}}}%
10175 }%

```

First use full form:

```

10176 \renewcommand*{\glxtrfullformat}[2]{%
10177   \glxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
10178 }%
10179 \renewcommand*{\glxtrfullplformat}[2]{%
10180   \glxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
10181 }%
10182 \renewcommand*{\Glsxtrfullformat}[2]{%
10183   \glxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
10184 }%
10185 \renewcommand*{\Glsxtrfullplformat}[2]{%
10186   \glxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
10187 }%

```

In-line format.

```

10188 \renewcommand*{\glxtrinlinefullformat}[2]{%
10189   \glfirstlonghyphenfont{\glsaccesslong{##1}}%
10190   \ifglxtrininsertinside{##2}\fi}%
10191   \ifglxtrininsertinside \else{##2}\fi
10192 }%
10193 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10194   \glfirstlonghyphenfont{\glsaccesslongpl{##1}}%
10195   \ifglxtrininsertinside{##2}\fi}%
10196   \ifglxtrininsertinside \else{##2}\fi
10197 }%
10198 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10199   \glfirstlonghyphenfont{\Glsaccesslong{##1}}%
10200   \ifglxtrininsertinside{##2}\fi}%
10201   \ifglxtrininsertinside \else{##2}\fi
10202 }%
10203 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10204   \glfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
10205   \ifglxtrininsertinside{##2}\fi}%
10206   \ifglxtrininsertinside \else{##2}\fi
10207 }%
10208 }

```

ort-hyphen-desc Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

10209 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
10210 {%
10211   \renewcommand*{\CustomAbbreviationFields}{%
10212     name={\glxstrlongshortdescname},
10213     sort={\glxstrlongshortdescsort},%
10214     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
10215     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
10216     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10217     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10218   }%
10219   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10220     \csdef{glxstrpostlink\glscategorylabel}{%
10221       \glxstrifwasfirstuse
10222       {%
10223         \glxstrposthyphenshort{\glslabel}{\glsinsert}%
10224       }%
10225     }%

```

Put the insertion into the post-link:

```

10226       \glxstrposthyphensubsequent{\glslabel}{\glsinsert}%
10227     }%
10228   }%
10229   \glshasattribute{\the\glslabeltok}{regular}%
10230   {%
10231     \glissetattribute{\the\glslabeltok}{regular}{false}%
10232   }%
10233   {}%
10234 }%
10235 }%
10236 {%
10237   \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
10238 }

```

rshorthyphenlong `\glxtrshorthyphenlong{<label>}{<short>}{<long>}{<insert>}`

The *<long>* and *<short>* arguments may be the plural form. The *<long>* argument may also be the first letter uppercase form.

```
10239 \newcommand*{\glxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
10240 {%
```

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.)

```
10241   \glxstrifhyphenstart{#4}{\def\glxtrwordsep{-}}{}%

```

```

10242 \glsfirstabbrvhyphenfont{#2\ifglxtrinsertinside{#4}\fi}%
10243 \ifglxtrinsertinside\else{#4}\fi
10244 \glxtrfullsep{#1}%
10245 \glxtrparen{\glsfirstlonghyphenfont{#3\ifglxtrinsertinside{#4}\fi}%
10246 \ifglxtrinsertinside\else{#4}\fi}%
10247 }%
10248 }

```

hen-long-hyphen Designed for use with the markwords attribute.

```

10249 \newabbreviationstyle{short-hyphen-long-hyphen}%
10250 {%
10251 \renewcommand*{\CustomAbbreviationFields}{%
10252 name={\glxtrshortlongname},
10253 sort={\the\glsshorttok},
10254 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10255 \protect\glxtrfullsep{\the\glslabeltok}%
10256 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10257 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10258 \protect\glxtrfullsep{\the\glslabeltok}%
10259 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10260 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}},%
10261 description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10262 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10263 \glshasattribute{\the\glslabeltok}{regular}%
10264 {%
10265 \glissetattribute{\the\glslabeltok}{regular}{false}%
10266 }%
10267 {}%
10268 }%
10269 }%
10270 {%
10271 \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
10272 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10273 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10274 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10275 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10276 \renewcommand*{\glxtrfullformat}[2]{%
10277 \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
10278 }%
10279 \renewcommand*{\glxtrfullplformat}[2]{%
10280 \glxtrshorthyphenlong{##1}%
10281 {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10282 }%
10283 \renewcommand*{\Glsxtrfullformat}[2]{%
10284 \glxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
10285 }%

```

```

10286 \renewcommand*{\Glsxtrfullplformat}[2]{%
10287   \glsxtrshorthyphenlong{##1}%
10288   {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
10289 }%
10290 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

10291 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
10292 {%
10293   \renewcommand*{\CustomAbbreviationFields}{%
10294     name={\glsxtrshortlongdescname},
10295     sort={\glsxtrshortlongdescsort},
10296     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
10297       \protect\glsxtrfullsep{\the\glslabeltok}%
10298       \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
10299     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
10300       \protect\glsxtrfullsep{\the\glslabeltok}%
10301       \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
10302     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10303     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
10304   }%

```

Unset the regular attribute if it has been set.

```

10305 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10306   \glsasattribute{\the\glslabeltok}{regular}%
10307   {%
10308     \glissetattribute{\the\glslabeltok}{regular}{false}%
10309   }%
10310   {}}%
10311 }%
10312 }%
10313 {%
10314   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
10315 }

```

lsxtrshorthyphen `\glsxtrshorthyphen{<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.

```

10316 \newcommand*{\glsxtrshorthyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

10317 {%
10318   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
10319   \glsfirstabbrvhyphenfont{#1}%
10320 }%
10321 }

```

trposthyphenlong `\glxtrposthyphenlong{<label>}{<insert>}`

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glxtrshorthyphenlong` but omits the *<short>* part. This always uses the singular long form.

```

10322 \newcommand*{\glxtrposthyphenlong}[2]{%
10323   {%
10324     \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{%
10325       \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
10326       \glxtrfullsep{#1}%
10327       \glxtrparen
10328       {\glsfirstlonghyphenfont{\glsentrylong{#1}\ifglxtrininsertinside{#2}\fi}%
10329       \ifglxtrininsertinside\else{#2}\fi
10330     }%
10331   }%
10332 }
```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

10333 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
10334 {%
10335   \renewcommand*{\CustomAbbreviationFields}{%
10336     name={\glxtrshortlongname},
10337     sort={\the\glsshorttok},
10338     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10339     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10340     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
10341     description={\protect\glslonghyphenfont{\the\glslongtok}}}%
10342 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10343   \csdef{glxtrpostlink\glscategorylabel}{%
10344     \glxtrifwasfirstuse
10345     {%
10346       \glxtrposthyphenlong{\glslabel}{\glsinsert}%
10347     }%
10348     {%
```

Put the insertion into the post-link:

```

10349       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10350     }%
10351   }%
10352   \glshasattribute{\the\glslabeltok}{regular}%
10353   {%
10354     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10355   }%
10356   {}%
10357 }%
10358 }%
10359 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```

10360 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
10361 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
10362 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
10363 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
10364 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

10365 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10366   \glsabbrvfont{\glsaccessshort{##1}}%
10367 }%
10368 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10369   \glsabbrvfont{\glsaccessshortpl{##1}}%
10370 }%
10371 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10372   \glsabbrvfont{\Glsaccessshort{##1}}%
10373 }%
10374 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10375   \glsabbrvfont{\Glsaccessshortpl{##1}}%
10376 }%

```

First use full form:

```

10377 \renewcommand*{\glxtrfullformat}[2]{%
10378   \glxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
10379 }%
10380 \renewcommand*{\glxtrfullplformat}[2]{%
10381   \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
10382 }%
10383 \renewcommand*{\Glsxtrfullformat}[2]{%
10384   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
10385 }%
10386 \renewcommand*{\Glsxtrfullplformat}[2]{%
10387   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
10388 }%

```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

10389 \renewcommand*{\glxtrinlinefullformat}[2]{%
10390   \glsfirstabbrvhyphenfont{\glsaccessshort{##1}}%
10391   \ifglxtrininsertinside{##2}\fi}%
10392   \ifglxtrininsertinside \else{##2}\fi
10393 }%
10394 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10395   \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}}%
10396   \ifglxtrininsertinside{##2}\fi}%
10397   \ifglxtrininsertinside \else{##2}\fi
10398 }%
10399 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10400   \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}}%
10401   \ifglxtrininsertinside{##2}\fi}%

```

```

10402   \ifglxtrinsertinside \else{##2}\fi
10403 }%
10404 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10405   \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}}%
10406   \ifglxtrinsertinside{##2}\fi}%
10407   \ifglxtrinsertinside \else{##2}\fi
10408 }%
10409 }

```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```

10410 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
10411 {%
10412   \renewcommand*{\CustomAbbreviationFields}{%
10413     name={\glxtrshortlongdescname},
10414     sort={\glxtrshortlongdescsort},%
10415     first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
10416     firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
10417     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
10418     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
10419 }%
10420 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10421   \csdef{glxtrpostlink\glscategorylabel}{%
10422     \glxtrifwasfirstuse
10423     {%
10424       \glxtrposthyphenlong{\glslabel}{\glsinsert}%
10425     }%
10426     {%

```

Put the insertion into the post-link:

```

10427       \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
10428     }%
10429   }%
10430   \glshasattribute{\the\glslabeltok}{regular}%
10431   {%
10432     \glissetattribute{\the\glslabeltok}{regular}{false}%
10433   }%
10434   {}%
10435 }%
10436 }%
10437 {%
10438   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
10439 }

```

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

```

10440 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%

```

stabbrvonlyfont

```
10441 \newcommand*{\glsfirstabbrvonlyfont}{\glsabbrvonlyfont}%
```

glslongonlyfont

```
10442 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%
```

rstlongonlyfont

```
10443 \newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
10444 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtronlyname The default name format for this style.

```
10445 \newcommand*{\glsxtronlyname}{%
```

```
10446   \protect\glsabbrvonlyfont{\the\glsshorttok}}%
```

```
10447 }
```

only-short-only

```
10448 \newabbreviationstyle{long-only-short-only}%
```

```
10449 {%
```

```
10450   \renewcommand*{\CustomAbbreviationFields}{%
```

```
10451     name={\glsxtronlyname},
```

```
10452     sort={\the\glsshorttok},
```

```
10453     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
```

```
10454     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
```

```
10455     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
```

```
10456     description={\protect\glslongonlyfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
10457   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
10458     \glshasattribute{\the\glslabeltok}{regular}}%
```

```
10459   {%
```

```
10460     \glssetattribute{\the\glslabeltok}{regular}{false}}%
```

```
10461   }%
```

```
10462   {}%
```

```
10463 }%
```

```
10464 }%
```

```
10465 {%
```

```
10466   \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtronlysuffix}%
```

```
10467   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
```

```
10468   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
```

```
10469   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
```

```
10470   \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%
```

The first use full form doesn't show the short form.

```
10471   \renewcommand*{\glsxtrfullformat}[2]{%
```

```
10472     \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}}%
```

```

10473 \ifglxtrinsertinside\else##2\fi
10474 }%
10475 \renewcommand*{\glxtrfullplformat}[2]{%
10476 \glsfirslongonlyfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10477 \ifglxtrinsertinside\else##2\fi
10478 }%
10479 \renewcommand*{\Glsxtrfullformat}[2]{%
10480 \glsfirslongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10481 \ifglxtrinsertinside\else##2\fi
10482 }%
10483 \renewcommand*{\Glsxtrfullplformat}[2]{%
10484 \glsfirslongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10485 \ifglxtrinsertinside\else##2\fi
10486 }%

```

The inline full form does show the short form.

```

10487 \renewcommand*{\glxtrinlinefullformat}[2]{%
10488 \glsfirslongonlyfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10489 \ifglxtrinsertinside\else##2\fi
10490 \glxtrfullsep{##1}%
10491 \glxtrparen{\protect\glsfirstabbrvonlyfont{\glssaccessshort{##1}}}%
10492 }%
10493 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10494 \glsfirslongonlyfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10495 \ifglxtrinsertinside\else##2\fi
10496 \glxtrfullsep{##1}%
10497 \glxtrparen{\protect\glsfirstabbrvonlyfont{\glssaccessshortpl{##1}}}%
10498 }%
10499 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10500 \glsfirslongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10501 \ifglxtrinsertinside\else##2\fi
10502 \glxtrfullsep{##1}%
10503 \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10504 }%
10505 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10506 \glsfirslongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10507 \ifglxtrinsertinside\else##2\fi
10508 \glxtrfullsep{##1}%
10509 \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
10510 }%
10511 }

```

xtronlydescsort

```

10512 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtronlydescname

```

10513 \newcommand*{\glxtronlydescname}{%
10514 \protect\glslongfont{\the\glslongtok}%
10515 }

```

short-only-desc

```
10516 \newabbreviationstyle{long-only-short-only-desc}%
10517 {%
10518   \renewcommand*{\CustomAbbreviationFields}{%
10519     name={\glstronlydescname},
10520     sort={\glstronlydescsort},%
10521     first={\protect\glstrfirstlongonlyfont{\the\glstrlongtok}},%
10522     firstplural={\protect\glstrfirstlongonlyfont{\the\glstrlongpltok}},%
10523     text={\protect\glstrabbrvonlyfont{\the\glstrshorttok}},%
10524     plural={\protect\glstrabbrvonlyfont{\the\glstrshortpltok}}}%
10525   }%
10526   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10527     \glshasattribute{\the\glstrlabeltok}{regular}%
10528     {%
10529       \glsssetattribute{\the\glstrlabeltok}{regular}{false}%
10530     }%
10531   }%
10532 }%
10533 }%
10534 {%
10535   \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
10536 }
```

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 `\glstryshort`, instead but this doesn't reflect the formatting since it doesn't include `\glstrfont`. 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 `\glstrshort` 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:

```
10537 \let\@glxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```
10538 \renewcommand*{\markright}[1]{%
10539   \glxtrmarkhook
10540   \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
10541   \glxtrrestoremarkhook
10542 }
```

`\markboth` Save original definition:

```
10543 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
10544 \renewcommand*{\markboth}[2]{%
10545   \glxtrmarkhook
10546   \@glxtr@org@markboth
10547   {\@glxtrinmark#1\@glxtrnotinmark}%
10548   {\@glxtrinmark#2\@glxtrnotinmark}%
10549   \glxtrrestoremarkhook
10550 }
```

Also do this for `\@starttoc`

`\@starttoc` Save original definition:

```
10551 \let\@glxtr@org@@starttoc\@starttoc
```

Redefine:

```
10552 \renewcommand*{\@starttoc}[1]{%
10553   \glxtrmarkhook
10554   \@glxtrinmark
10555   \@glxtr@org@@starttoc{#1}%
10556   \@glxtrnotinmark
10557   \glxtrrestoremarkhook
10558 }
```

If this causes a problem provide a simple way of switching back to the original definitions:

`sxtrRevertMarks`

```
10559 \newcommand*{\glxtrRevertMarks}{%
10560   \let\markright\@glxtr@org@markright
10561   \let\markboth\@glxtr@org@markboth
10562   \let\@starttoc\@glxtr@org@@starttoc
10563 }
```

`\glxtrifinmark`

```
10564 \newcommand*{\glxtrifinmark}[2]{#2}
```

```

\@glsxtrinmark
10565 \newrobustcmd*{\@glsxtrinmark}{%
10566   \let\glsxtrifinmark\@firstoftwo
10567 }

glsxtrnotinmark
10568 \newrobustcmd*{\@glsxtrnotinmark}{%
10569   \let\glsxtrifinmark\@secondoftwo
10570 }

eorpdpforheading
10571 \ifdef\texorpdfstring
10572 {
10573   \newcommand*{\glsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
10574 }
10575 {
10576   \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
10577 }

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply
to the marks:
10578 \newcommand*{\glsxtrmarkhook}{%
  Save current definitions:
10579   \let\@glsxtr@org@MakeUppercase\MakeUppercase
10580   \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
10581   \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
10582   \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
10583   \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
10584   \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
10585   \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
10586   \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
10587   \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
10588   \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
10589   \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
10590   \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
10591   \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
10592   \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
10593   \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
10594   \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
10595   \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
10596   \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
10597   \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
10598   \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
10599   \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
10600   \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
10601   \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
10602   \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl

```

New definitions

```

10603 \let\glsxtrifinmark\@firstoftwo
10604 \let\MakeUppercase\MakeTextUppercase
10605 \let\glsxtrtitleorpdforheading\@thirdofthree
10606 \let\glsxtrtitleshort\glsxtrheadshort
10607 \let\glsxtrtitleshortpl\glsxtrheadshortpl
10608 \let\Glsxtrtitleshort\Glsxtrheadshort
10609 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
10610 \let\glsxtrtitlename\glsxtrheadname
10611 \let\Glsxtrtitlename\Glsxtrheadname
10612 \let\glsxtrtitletext\glsxtrheadtext
10613 \let\Glsxtrtitletext\Glsxtrheadtext
10614 \let\glsxtrtitleplural\glsxtrheadplural
10615 \let\Glsxtrtitleplural\Glsxtrheadplural
10616 \let\glsxtrtitlefirst\glsxtrheadfirst
10617 \let\Glsxtrtitlefirst\Glsxtrheadfirst
10618 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
10619 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
10620 \let\glsxtrtitlelong\glsxtrheadlong
10621 \let\glsxtrtitlelongpl\glsxtrheadlongpl
10622 \let\Glsxtrtitlelong\Glsxtrheadlong
10623 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
10624 \let\glsxtrtitlefull\glsxtrheadfull
10625 \let\glsxtrtitlefullpl\glsxtrheadfullpl
10626 \let\Glsxtrtitlefull\Glsxtrheadfull
10627 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
10628 }

```

`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.)

```

10629 \newcommand*{\glsxtrrestoremarkhook}{%
10630 \let\glsxtrifinmark\@secondoftwo
10631 \let\MakeUppercase\@glsxtr@org@MakeUppercase
10632 \let\glsxtrtitleorpdforheading\@glsxtr@org@glsxtrtitleorpdforheading
10633 \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
10634 \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
10635 \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
10636 \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
10637 \let\glsxtrtitlename\@glsxtr@org@glsxtrtitlename
10638 \let\Glsxtrtitlename\@glsxtr@org@Glsxtrtitlename
10639 \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
10640 \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
10641 \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
10642 \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
10643 \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
10644 \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
10645 \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural

```

```

10646 \let\Glsxtrtitlefirstplural\@glxtr@org@Glsxtrtitlefirstplural
10647 \let\glxtrtitlelong\@glxtr@org@glxtrtitlelong
10648 \let\glxtrtitlelongpl\@glxtr@org@glxtrtitlelongpl
10649 \let\Glsxtrtitlelong\@glxtr@org@Glsxtrtitlelong
10650 \let\Glsxtrtitlelongpl\@glxtr@org@Glsxtrtitlelongpl
10651 \let\glxtrtitlefull\@glxtr@org@glxtrtitlefull
10652 \let\glxtrtitlefullpl\@glxtr@org@glxtrtitlefullpl
10653 \let\Glsxtrtitlefull\@glxtr@org@Glsxtrtitlefull
10654 \let\Glsxtrtitlefullpl\@glxtr@org@Glsxtrtitlefullpl
10655 }

```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

glxtrheadshort Command used to display short form in the page header.

```

10656 \newcommand*{\glxtrheadshort}[1]{%
10657   \protect\NoCaseChange
10658   {%
10659     \gl@ifattribute{#1}{headuc}{true}%
10660     {%
10661       \GLSxtrshort[noindex,hyper=false]{#1}[]%
10662     }%
10663     {%
10664       \glxtrshort[noindex,hyper=false]{#1}[]%
10665     }%
10666   }%
10667 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

10668 \newrobustcmd*{\lsxtrtitleshort}[1]{%
10669   \glxtrshort[noindex,hyper=false]{#1}[]%
10670 }

```

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.

```

10671 \newcommand*{\glxtrheadshortpl}[1]{%
10672   \protect\NoCaseChange
10673   {%
10674     \gl@ifattribute{#1}{headuc}{true}%
10675     {%
10676       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
10677     }%
10678     {%
10679       \glxtrshortpl[noindex,hyper=false]{#1}[]%
10680     }%
10681   }%
10682 }

```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```
10683 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
10684   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
10685 }
```

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```
10686 \newcommand*{\Glsxtrheadshort}[1]{%
10687   \protect\NoCaseChange
10688   {%
10689     \glsifattribute{#1}{headuc}{true}%
10690     {%
10691       \GLSxtrshort[noindex,hyper=false]{#1}[]%
10692     }%
10693     {%
10694       \Glsxtrshort[noindex,hyper=false]{#1}[]%
10695     }%
10696   }%
10697 }
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
10698 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
10699   \Glsxtrshort[noindex,hyper=false]{#1}[]%
10700 }
```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```
10701 \newcommand*{\Glsxtrheadshortpl}[1]{%
10702   \protect\NoCaseChange
10703   {%
10704     \glsifattribute{#1}{headuc}{true}%
10705     {%
10706       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
10707     }%
10708     {%
10709       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
10710     }%
10711   }%
10712 }
```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
10713 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
10714   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
10715 }
```

`\Glsxtrheadname` As above but for the name value.

```

10716 \newcommand*{\glxtrheadname}[1]{%
10717   \protect\NoCaseChange
10718   {%
10719     \gl@ifattribute{#1}{headuc}{true}%
10720     {%
10721       \GLSname[noindex,hyper=false]{#1}[]%
10722     }%
10723     {%
10724       \glSname[noindex,hyper=false]{#1}[]%
10725     }%
10726   }%
10727 }

```

glxstrtitlename Command to display name value in section title and table of contents.

```

10728 \newrobustcmd*{\glxstrtitlename}[1]{%
10729   \glSname[noindex,hyper=false]{#1}[]%
10730 }

```

\Glsxtrheadname First letter converted to upper case

```

10731 \newcommand*{\Glsxtrheadname}[1]{%
10732   \protect\NoCaseChange
10733   {%
10734     \gl@ifattribute{#1}{headuc}{true}%
10735     {%
10736       \GLSname[noindex,hyper=false]{#1}[]%
10737     }%
10738     {%
10739       \glSname[noindex,hyper=false]{#1}[]%
10740     }%
10741   }%
10742 }

```

Glsxtrtitlename Command to display name value in section title and table of contents with the first letter changed to upper case.

```

10743 %\changes{1.21}{2017-11-03}{new}
10744 \newrobustcmd*{\Glsxtrtitlename}[1]{%
10745   \glSname[noindex,hyper=false]{#1}[]%
10746 }

```

\glxtrheadtext As above but for the text value.

```

10747 \newcommand*{\glxtrheadtext}[1]{%
10748   \protect\NoCaseChange
10749   {%
10750     \gl@ifattribute{#1}{headuc}{true}%
10751     {%
10752       \GLStext[noindex,hyper=false]{#1}[]%
10753     }%
10754     {%
10755       \glstext[noindex,hyper=false]{#1}[]%

```

```

10756 }%
10757 }%
10758 }

```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```

10759 \newrobustcmd*{\glsxtrtitletext}[1]{%
10760   \glstext[noindex,hyper=false]{#1}[]%
10761 }

```

`\Glsxtrheadtext` First letter converted to upper case

```

10762 \newcommand*{\Glsxtrheadtext}[1]{%
10763   \protect\NoCaseChange
10764   {%
10765     \glsifattribute{#1}{headuc}{true}%
10766     {%
10767       \GLStext[noindex,hyper=false]{#1}[]%
10768     }%
10769     {%
10770       \Glstext[noindex,hyper=false]{#1}[]%
10771     }%
10772   }%
10773 }

```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```

10774 \newrobustcmd*{\Glsxtrtitletext}[1]{%
10775   \Glstext[noindex,hyper=false]{#1}[]%
10776 }

```

`lgsxtrheadplural` As above but for the plural value.

```

10777 \newcommand*{\lgsxtrheadplural}[1]{%
10778   \protect\NoCaseChange
10779   {%
10780     \glsifattribute{#1}{headuc}{true}%
10781     {%
10782       \GLSplural[noindex,hyper=false]{#1}[]%
10783     }%
10784     {%
10785       \glsplural[noindex,hyper=false]{#1}[]%
10786     }%
10787   }%
10788 }

```

`sxtrtitleplural` Command to display plural value in section title and table of contents.

```

10789 \newrobustcmd*{\sxtrtitleplural}[1]{%
10790   \glsplural[noindex,hyper=false]{#1}[]%
10791 }

```

lsxtrheadplural Convert first letter to upper case.

```
10792 \newcommand*{\Glsxtrheadplural}[1]{%
10793   \protect\NoCaseChange
10794   {%
10795     \glsifattribute{#1}{headuc}{true}%
10796     {%
10797       \GLSplural[noindex,hyper=false]{#1}[]%
10798     }%
10799     {%
10800       \Glsplural[noindex,hyper=false]{#1}[]%
10801     }%
10802   }%
10803 }
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
10804 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
10805   \Glsplural[noindex,hyper=false]{#1}[]%
10806 }
```

glxtrheadfirst As above but for the first value.

```
10807 \newcommand*{\glxtrheadfirst}[1]{%
10808   \protect\NoCaseChange
10809   {%
10810     \glsifattribute{#1}{headuc}{true}%
10811     {%
10812       \GLSfirst[noindex,hyper=false]{#1}[]%
10813     }%
10814     {%
10815       \glsfirst[noindex,hyper=false]{#1}[]%
10816     }%
10817   }%
10818 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
10819 \newrobustcmd*{\glxtrtitlefirst}[1]{%
10820   \glsfirst[noindex,hyper=false]{#1}[]%
10821 }
```

Glsxtrheadfirst First letter converted to upper case

```
10822 \newcommand*{\Glsxtrheadfirst}[1]{%
10823   \protect\NoCaseChange
10824   {%
10825     \glsifattribute{#1}{headuc}{true}%
10826     {%
10827       \GLSfirst[noindex,hyper=false]{#1}[]%
10828     }%
10829     {%
```

```

10830      \Glsfirst[noindex,hyper=false]{#1}[]%
10831    }%
10832  }%
10833 }

```

`\lsxtrtitlefirst` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

10834 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
10835   \Glsfirst[noindex,hyper=false]{#1}[]%
10836 }

```

`\headfirstplural` As above but for the firstplural value.

```

10837 \newcommand*{\Glsxtrheadfirstplural}[1]{%
10838   \protect\NoCaseChange
10839   {%
10840     \glsifattribute{#1}{headuc}{true}%
10841     {%
10842       \GLSfirstplural[noindex,hyper=false]{#1}[]%
10843     }%
10844     {%
10845       \glsfirstplural[noindex,hyper=false]{#1}[]%
10846     }%
10847   }%
10848 }

```

`\itlefirstplural` Command to display firstplural value in section title and table of contents.

```

10849 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
10850   \glsfirstplural[noindex,hyper=false]{#1}[]%
10851 }

```

`\headfirstplural` First letter converted to upper case

```

10852 \newcommand*{\Glsxtrheadfirstplural}[1]{%
10853   \protect\NoCaseChange
10854   {%
10855     \glsifattribute{#1}{headuc}{true}%
10856     {%
10857       \GLSfirstplural[noindex,hyper=false]{#1}[]%
10858     }%
10859     {%
10860       \Glsfirstplural[noindex,hyper=false]{#1}[]%
10861     }%
10862   }%
10863 }

```

`\itlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

10864 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
10865   \Glsfirstplural[noindex,hyper=false]{#1}[]%
10866 }

```

`\glxstrheadlong` Command used to display long form in the page header.

```
10867 \newcommand*{\glxstrheadlong}[1]{%
10868   \protect\NoCaseChange
10869   {%
10870     \gl@ifattribute{#1}{headuc}{true}%
10871     {%
10872       \GLSxtrlong[noindex,hyper=false]{#1}[]%
10873     }%
10874     {%
10875       \glxstrlong[noindex,hyper=false]{#1}[]%
10876     }%
10877   }%
10878 }
```

`glxstrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
10879 \newrobustcmd*{\glxstrtitlelong}[1]{%
10880   \glxstrlong[noindex,hyper=false]{#1}[]%
10881 }
```

`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.

```
10882 \newcommand*{\glxtrheadlongpl}[1]{%
10883   \protect\NoCaseChange
10884   {%
10885     \gl@ifattribute{#1}{headuc}{true}%
10886     {%
10887       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
10888     }%
10889     {%
10890       \glxstrlongpl[noindex,hyper=false]{#1}[]%
10891     }%
10892   }%
10893 }
```

`sxstrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
10894 \newrobustcmd*{\glxstrtitlelongpl}[1]{%
10895   \glxstrlongpl[noindex,hyper=false]{#1}[]%
10896 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```
10897 \newcommand*{\Glsxtrheadlong}[1]{%
10898   \protect\NoCaseChange
10899   {%
10900     \gl@ifattribute{#1}{headuc}{true}%
10901     {%
10902       \GLSxtrlong[noindex,hyper=false]{#1}[]%
```

```

10903 }%
10904 {%
10905     \Glsxtrlong[noindex,hyper=false]{#1}[]%
10906 }%
10907 }%
10908 }

```

Glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

10909 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
10910     \Glsxtrlong[noindex,hyper=false]{#1}[]%
10911 }

```

lgsxtrheadlongpl Command used to display plural long form in the page header with the first letter converted to upper case.

```

10912 \newcommand*{\Glsxtrheadlongpl}[1]{%
10913     \protect\NoCaseChange
10914     {%
10915         \glsifattribute{#1}{headuc}{true}%
10916         {%
10917             \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
10918         }%
10919         {%
10920             \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
10921         }%
10922     }%
10923 }

```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

10924 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
10925     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
10926 }

```

\glsxtrheadfull Command used to display full form in the page header.

```

10927 \newcommand*{\glsxtrheadfull}[1]{%
10928     \protect\NoCaseChange
10929     {%
10930         \glsifattribute{#1}{headuc}{true}%
10931         {%
10932             \GLSxtrfull[noindex,hyper=false]{#1}[]%
10933         }%
10934         {%
10935             \glsxtrfull[noindex,hyper=false]{#1}[]%
10936         }%
10937     }%
10938 }

```

`glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```
10939 \newrobustcmd*{\glsxtrtitlefull}[1]{%
10940   \glsxtrfull[noindex,hyper=false]{#1}[]%
10941 }
```

`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.

```
10942 \newcommand*{\glsxtrheadfullpl}[1]{%
10943   \protect\NoCaseChange
10944   {%
10945     \glsifattribute{#1}{headuc}{true}%
10946     {%
10947       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
10948     }%
10949     {%
10950       \glsxtrfullpl[noindex,hyper=false]{#1}[]%
10951     }%
10952   }%
10953 }
```

`sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents.

```
10954 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%
10955   \glsxtrfullpl[noindex,hyper=false]{#1}[]%
10956 }
```

`\Glsxtrheadfull` Command used to display full form in the page header with the first letter converted to upper case.

```
10957 \newcommand*{\Glsxtrheadfull}[1]{%
10958   \protect\NoCaseChange
10959   {%
10960     \glsifattribute{#1}{headuc}{true}%
10961     {%
10962       \GLSxtrfull[noindex,hyper=false]{#1}[]%
10963     }%
10964     {%
10965       \Glsxtrfull[noindex,hyper=false]{#1}[]%
10966     }%
10967   }%
10968 }
```

`Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
10969 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
10970   \Glsxtrfull[noindex,hyper=false]{#1}[]%
10971 }
```

`\lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

10972 \newcommand*{\Glsxtrheadfullpl}[1]{%
10973   \protect\NoCaseChange
10974   {%
10975     \glsifattribute{#1}{headuc}{true}%
10976     {%
10977       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
10978     }%
10979     {%
10980       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
10981     }%
10982   }%
10983 }

```

`\lsxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

10984 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
10985   \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
10986 }

```

`\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.

```

10987 \ifdef\texorpdfstring
10988 {
10989   \newcommand*{\glsfmtshort}[1]{%
10990     \texorpdfstring
10991       {\glsxtrtitleshort{#1}}%
10992       {\glsentryshort{#1}}%
10993   }
10994 }
10995 {
10996   \newcommand*{\glsfmtshort}[1]{%
10997     \glsxtrtitleshort{#1}}
10998 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

10999 \ifdef\texorpdfstring
11000 {
11001   \newcommand*{\glsfmtshortpl}[1]{%
11002     \texorpdfstring
11003       {\glsxtrtitleshortpl{#1}}%
11004       {\glsentryshortpl{#1}}%
11005   }
11006 }
11007 {
11008   \newcommand*{\glsfmtshortpl}[1]{%

```

```

11009 \glstrtitleshortpl{#1}}
11010 }

```

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).

```

11011 \ifdef\texorpdfstring
11012 {
11013   \newcommand*\Glsfmtshort}[1]{%
11014     \texorpdfstring
11015       {\Glsxtrtitleshort{#1}}%
11016       {\glstryshort{#1}}%
11017   }
11018 }
11019 {
11020   \newcommand*\Glsfmtshort}[1]{%
11021     \Glsxtrtitleshort{#1}}
11022 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

11023 \ifdef\texorpdfstring
11024 {
11025   \newcommand*\Glsfmtshortpl}[1]{%
11026     \texorpdfstring
11027       {\Glsxtrtitleshortpl{#1}}%
11028       {\glstryshortpl{#1}}%
11029   }
11030 }
11031 {
11032   \newcommand*\Glsfmtshortpl}[1]{%
11033     \Glsxtrtitleshortpl{#1}}
11034 }

```

`\glsfmtname` As above but for the name value.

```

11035 \ifdef\texorpdfstring
11036 {
11037   \newcommand*\glsfmtname}[1]{%
11038     \texorpdfstring
11039       {\glxtrtitlename{#1}}%
11040       {\glstryname{#1}}%
11041   }
11042 }
11043 {
11044   \newcommand*\glsfmtname}[1]{%
11045     \glxtrtitlename{#1}}
11046 }

```

`\Glsfmtname` First letter converted to upper case.

```

11047 \ifdef\texorpdfstring
11048 {
11049   \newcommand*{\Glsfmtname}[1]{%
11050     \texorpdfstring
11051     {\Glsxtrtitlename{#1}}%
11052     {\glsentryname{#1}}%
11053   }
11054 }
11055 {
11056   \newcommand*{\Glsfmtname}[1]{%
11057     \Glsxtrtitlename{#1}}
11058 }

```

`\glsfmttext` As above but for the text value.

```

11059 \ifdef\texorpdfstring
11060 {
11061   \newcommand*{\glsfmttext}[1]{%
11062     \texorpdfstring
11063     {\glsxtrtitletext{#1}}%
11064     {\glsentrytext{#1}}%
11065   }
11066 }
11067 {
11068   \newcommand*{\glsfmttext}[1]{%
11069     \glsxtrtitletext{#1}}
11070 }

```

`\Glsfmttext` First letter converted to upper case.

```

11071 \ifdef\texorpdfstring
11072 {
11073   \newcommand*{\Glsfmttext}[1]{%
11074     \texorpdfstring
11075     {\Glsxtrtitletext{#1}}%
11076     {\glsentrytext{#1}}%
11077   }
11078 }
11079 {
11080   \newcommand*{\Glsfmttext}[1]{%
11081     \Glsxtrtitletext{#1}}
11082 }

```

`\glsfmtplural` As above but for the plural value.

```

11083 \ifdef\texorpdfstring
11084 {
11085   \newcommand*{\glsfmtplural}[1]{%
11086     \texorpdfstring
11087     {\glsxtrtitleplural{#1}}%
11088     {\glsentryplural{#1}}%
11089   }

```

```

11090 }
11091 {
11092   \newcommand*{\glsfmtplural}[1]{%
11093     \glsxtrtitleplural{#1}}
11094 }

```

`\Glsfmtplural` First letter converted to upper case.

```

11095 \ifdef\texorpdfstring
11096 {
11097   \newcommand*{\Glsfmtplural}[1]{%
11098     \texorpdfstring
11099     {\Glsxtrtitleplural{#1}}%
11100     {\glsentryplural{#1}}%
11101   }
11102 }
11103 {
11104   \newcommand*{\Glsfmtplural}[1]{%
11105     \Glsxtrtitleplural{#1}}
11106 }

```

`\glsfmtfirst` As above but for the first value.

```

11107 \ifdef\texorpdfstring
11108 {
11109   \newcommand*{\glsfmtfirst}[1]{%
11110     \texorpdfstring
11111     {\glsxtrtitlefirst{#1}}%
11112     {\glsentryfirst{#1}}%
11113   }
11114 }
11115 {
11116   \newcommand*{\glsfmtfirst}[1]{%
11117     \glsxtrtitlefirst{#1}}
11118 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

11119 \ifdef\texorpdfstring
11120 {
11121   \newcommand*{\Glsfmtfirst}[1]{%
11122     \texorpdfstring
11123     {\Glsxtrtitlefirst{#1}}%
11124     {\glsentryfirst{#1}}%
11125   }
11126 }
11127 {
11128   \newcommand*{\Glsfmtfirst}[1]{%
11129     \Glsxtrtitlefirst{#1}}
11130 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

11131 \ifdef\texorpdfstring
11132 {
11133   \newcommand*{\glsfmtfirstpl}[1]{%
11134     \texorpdfstring
11135     {\glsxtrtitlefirstplural{#1}}%
11136     {\glsentryfirstplural{#1}}%
11137   }
11138 }
11139 {
11140   \newcommand*{\glsfmtfirstpl}[1]{%
11141     \glsxtrtitlefirstplural{#1}}
11142 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

11143 \ifdef\texorpdfstring
11144 {
11145   \newcommand*{\Glsfmtfirstpl}[1]{%
11146     \texorpdfstring
11147     {\Glsxtrtitlefirstplural{#1}}%
11148     {\glsentryfirstplural{#1}}%
11149   }
11150 }
11151 {
11152   \newcommand*{\Glsfmtfirstpl}[1]{%
11153     \Glsxtrtitlefirstplural{#1}}
11154 }

```

`\glsfmtlong` As above but for the long value.

```

11155 \ifdef\texorpdfstring
11156 {
11157   \newcommand*{\glsfmtlong}[1]{%
11158     \texorpdfstring
11159     {\glsxtrtitlelong{#1}}%
11160     {\glsentrylong{#1}}%
11161   }
11162 }
11163 {
11164   \newcommand*{\glsfmtlong}[1]{%
11165     \glsxtrtitlelong{#1}}
11166 }

```

`\Glsfmtlong` First letter converted to upper case.

```

11167 \ifdef\texorpdfstring
11168 {
11169   \newcommand*{\Glsfmtlong}[1]{%
11170     \texorpdfstring
11171     {\Glsxtrtitlelong{#1}}%
11172     {\glsentrylong{#1}}%
11173   }

```

```

11174 }
11175 {
11176   \newcommand*{\Glsfmtlong}[1]{%
11177     \Glsxtrtitlelong{#1}}
11178 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

11179 \ifdef\texorpdfstring
11180 {
11181   \newcommand*{\glsfmtlongpl}[1]{%
11182     \texorpdfstring
11183     {\Glsxtrtitlelongpl{#1}}%
11184     {\Glsentrylongpl{#1}}%
11185   }
11186 }
11187 {
11188   \newcommand*{\glsfmtlongpl}[1]{%
11189     \Glsxtrtitlelongpl{#1}}
11190 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

11191 \ifdef\texorpdfstring
11192 {
11193   \newcommand*{\Glsfmtlongpl}[1]{%
11194     \texorpdfstring
11195     {\Glsxtrtitlelongpl{#1}}%
11196     {\Glsentrylongpl{#1}}%
11197   }
11198 }
11199 {
11200   \newcommand*{\Glsfmtlongpl}[1]{%
11201     \Glsxtrtitlelongpl{#1}}
11202 }

```

`\glsfmtfull` In-line full format.

```

11203 \ifdef\texorpdfstring
11204 {
11205   \newcommand*{\glsfmtfull}[1]{%
11206     \texorpdfstring
11207     {\Glsxtrtitlefull{#1}}%
11208     {\Glsxtrinlinefullformat{#1}{}}%
11209   }
11210 }
11211 {
11212   \newcommand*{\glsfmtfull}[1]{%
11213     \Glsxtrtitlefull{#1}}
11214 }

```

`\Glsfmtfull` First letter converted to upper case.

```

11215 \ifdef\txorpdfstring
11216 {
11217   \newcommand*{\Glsfmtfull}[1]{%
11218     \txorpdfstring
11219     {\Glsxtrtitlefull{#1}}%
11220     {\Glsxtrinlinefullformat{#1}{}}%
11221   }
11222 }
11223 {
11224   \newcommand*{\Glsfmtfull}[1]{%
11225     \Glsxtrtitlefull{#1}}
11226 }

```

`\glsfmtfullpl` In-line full plural format.

```

11227 \ifdef\txorpdfstring
11228 {
11229   \newcommand*{\glsfmtfullpl}[1]{%
11230     \txorpdfstring
11231     {\glsxtrtitlefullpl{#1}}%
11232     {\glsxtrinlinefullplformat{#1}{}}%
11233   }
11234 }
11235 {
11236   \newcommand*{\glsfmtfullpl}[1]{%
11237     \glsxtrtitlefullpl{#1}}
11238 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

11239 \ifdef\txorpdfstring
11240 {
11241   \newcommand*{\Glsfmtfullpl}[1]{%
11242     \txorpdfstring
11243     {\Glsxtrtitlefullpl{#1}}%
11244     {\Glsxtrinlinefullplformat{#1}{}}%
11245   }
11246 }
11247 {
11248   \newcommand*{\Glsfmtfullpl}[1]{%
11249     \Glsxtrtitlefullpl{#1}}
11250 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

11251 \newcommand*{\RequireGlossariesExtraLang}[1]{%

```

```

11252 \ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
11253 }

```

sariesExtraLang

```

11254 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
11255   \ProvidesFile{glossariesxtr-#1.ldf}%
11256 }

```

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.

```

11257 \newcommand{\glxtr@loaddialect}{%
11258   \IfTrackedLanguageFileExists{\this@dialect}%
11259   {glossariesxtr-}% prefix
11260   {.ldf}%
11261   {%
11262     \RequireGlossariesExtraLang{\CurrentTrackedTag}%
11263   }%
11264   {}% not found

```

If `glossaries-extra-bib2gls` has been loaded, `\@glxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```

11265 \@glxtrdialecthook
11266 }

```

```

11267 \ifpackageloaded{tracklang}
11268 {%
11269   \AnyTrackedLanguages
11270   {%
11271     \ForEachTrackedDialect{\this@dialect}{\glxtr@loaddialect}%
11272   }%
11273   {}%
11274 }
11275 {}

```

Load `glossaries-extra-stylemods` if required.

```

11276 \@glxtr@redefstyles

```

and set the style:

```

11277 \@glxtr@do@style

```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the `record` option.

```

11278 \NeedsTeXFormat{LaTeX2e}
11279 \ProvidesPackage{glossaries-extra-bib2gls}[2018/04/09 v1.29 (NLCT)]

```

These are some convenient macros for use with custom rules.

`\glshex`

```

11280 \newcommand*{\glshex}{\string\u}

```

`\providecommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in `bib2gls`.

```

11281 \newcommand*{\glsxtrprovidecommand}{\providecommand}

```

`\glossarylocation` For use with `indexcounter` and `bib2gls`.

```

11282 \newcommand*{\glsxtr@wrglossarylocation}[2]{#1}

```

`\IndexCounterLink`

```
\GlsXtrIndexCounterLink{<text>}{<label>}
```

For use with `indexcounter` and `bib2gls`.

```

11283 \ifdef\hyperref
11284 {%
11285   \newcommand*{\GlsXtrIndexCounterLink}[2]{%
11286     \glsxtrifhasfield{indexcounter}{#2}%
11287     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
11288     {#1}%
11289   }
11290 }
11291 {
11292   \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
11293 }

```

`\TeXEntryAliases` Convenient shortcut for use with `entry-type-aliases` to alias standard Bib_T_EX entry types to `@bibtexentry`.

```

11294 \newcommand*{\GlsXtrBibTeXEntryAliases}{%
11295   article=bibtexentry,
11296   book=bibtexentry,
11297   booklet=bibtexentry,
11298   conference=bibtexentry,
11299   inbook=bibtexentry,
11300   incollection=bibtexentry,
11301   inproceedings=bibtexentry,
11302   manual=bibtexentry,
11303   mastersthesis=bibtexentry,
11304   misc=bibtexentry,
11305   phdthesis=bibtexentry,
11306   proceedings=bibtexentry,
11307   techreport=bibtexentry,

```

```
11308 unpublished=bibtexentry
11309 }
```

ideBibTeXFields Convenient shortcut to define the standard Bib_T_EX fields.

```
11310 \newcommand*{\GlsXtrProvideBibTeXFields}{%
11311   \glsaddstoragekey{address}{}{\glstrbibaddress}%
11312   \glsaddstoragekey{author}{}{\glstrbibauthor}%
11313   \glsaddstoragekey{booktitle}{}{\glstrbibbooktitle}%
11314   \glsaddstoragekey{chapter}{}{\glstrbibchapter}%
11315   \glsaddstoragekey{edition}{}{\glstrbibedition}%
11316   \glsaddstoragekey{howpublished}{}{\glstrbibhowpublished}%
11317   \glsaddstoragekey{institution}{}{\glstrbibinstitution}%
11318   \glsaddstoragekey{journal}{}{\glstrbibjournal}%
11319   \glsaddstoragekey{month}{}{\glstrbibmonth}%
11320   \glsaddstoragekey{note}{}{\glstrbibnote}%
11321   \glsaddstoragekey{number}{}{\glstrbibnumber}%
11322   \glsaddstoragekey{organization}{}{\glstrbiborganization}%
11323   \glsaddstoragekey{pages}{}{\glstrbibpages}%
11324   \glsaddstoragekey{publisher}{}{\glstrbibpublisher}%
11325   \glsaddstoragekey{school}{}{\glstrbibschooll}%
11326   \glsaddstoragekey{series}{}{\glstrbibseries}%
11327   \glsaddstoragekey{title}{}{\glstrbibtitle}%
11328   \glsaddstoragekey{bibtexkey}{}{\glstrbibtype}%
11329   \glsaddstoragekey{volume}{}{\glstrbibvolume}%
11330 }
```

Provide missing Greek letters for use in maths mode. These commands are recognised by bib2gls and will be mapped to the Mathematical Greek Italic letters. This ensures that the Greek letters that have the same shape as Latin letters are kept with the other mathematical Greek letters for sorting purposes. The \LaTeX version of these commands (provided here) use an upright font for capitals and italic for lower case to provide a better match with the other Greek symbols provided by the kernel.

\backslash Alpha

```
11331 \providecommand*{\Alpha}{\mathrm{A}}
```

\backslash Beta

```
11332 \providecommand*{\Beta}{\mathrm{B}}
```

\backslash Epsilon

```
11333 \providecommand*{\Epsilon}{\mathrm{E}}
```

\backslash Zeta

```
11334 \providecommand*{\Zeta}{\mathrm{Z}}
```

\backslash Eta

```
11335 \providecommand*{\Eta}{\mathrm{H}}
```

```

\Iota
11336 \providecommand*\Iota{\mathrm{I}}

\Kappa
11337 \providecommand*\Kappa{\mathrm{K}}

\Mu
11338 \providecommand*\Mu{\mathrm{M}}

\Nu
11339 \providecommand*\Nu{\mathrm{N}}

\Omicron
11340 \providecommand*\Omicron{\mathrm{O}}

\Rho
11341 \providecommand*\Rho{\mathrm{P}}

\Tau
11342 \providecommand*\Tau{\mathrm{T}}

\Chi
11343 \providecommand*\Chi{\mathrm{X}}

\Digamma
11344 \providecommand*\Digamma{\mathrm{F}}

\omicron
11345 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
11346 \@ifpackageloaded{upgreek}%
11347 {

\Upalpha
11348   \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
11349   \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
11350   \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
11351   \providecommand*\Upzeta{\mathrm{Z}}

```

```

\Upeta
11352 \providecommand*\Upeta{\mathrm{H}}

\Upiota
11353 \providecommand*\Upiota{\mathrm{I}}

\Upkappa
11354 \providecommand*\Upkappa{\mathrm{K}}

\Upmu
11355 \providecommand*\Upmu{\mathrm{M}}

\Upnu
11356 \providecommand*\Upnu{\mathrm{N}}

\Upomicron
11357 \providecommand*\Upomicron{\mathrm{O}}

\Uprho
11358 \providecommand*\Uprho{\mathrm{P}}

\Uptau
11359 \providecommand*\Uptau{\mathrm{T}}

\Upchi
11360 \providecommand*\Upchi{\mathrm{X}}

\upomicron
11361 \providecommand*\upomicron{\mathrm{o}}

11362 }%
11363 {}% upgreek.sty not loaded

```

This package provides some basic rules, but it's not intended for complete coverage of all locales. The CLDR should provide the appropriate locale-sensitive rules. These macros are primarily to help construct custom rules to include, for example, Greek maths symbols mixed with Latin. For the full rule syntax, see the Java API for [RuleBaseCollator](#)

If you want to provide a rule-block for a particular locale to allow for customization within that locale, create a file called `glossariesxtr-<tag>.ldf` (where *<tag>* identifies the locale) and add similar commands. See the description of `\IfTrackedLanguageFileExists` in the `tracklang` manual for the allowed forms of *<tag>*. The simplest is to just use the root language label or ISO code. The file will then be automatically loaded by `glossaries-extra` if the document has support for that language.

When combining these blocks of rules, remember to separate them with the appropriate character. For example:

```
sort-rule={\glxtrcontrolrules
```

```

;\glxtrspacerules
;\glxtrnonprintablerules
;\glxtrcombiningdiacriticrules
,\glxtrhyphenrules
<\glxtrgeneralpuncrules
<\glxtrdigitrules
<\glxtrfractionrules
<\glxtrGeneralLatinIVrules
<\glxtrMathItalicGreekIrules
}

```

xtrcontrolrules These are control characters that are usually placed at the start of a rule in the ‘ignored characters’ section. These control characters are unlikely to appear in any entry fields but are provided for completeness. \string is used for punctuation characters in case they’ve been made active.

```

11364 \newcommand*{\glxtrcontrolrules}{%
11365 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
11366 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
11367 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
11368 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
11369 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
11370 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
11371 0010\string'\string=\glshex 0011
11372 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
11373 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
11374 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
11375 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
11376 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
11377 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
11378 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
11379 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
11380 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
11381 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
11382 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
11383 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
11384 }

```

lsxtrspacerules These are space characters.

```

11385 \newcommand*{\glxtrspacerules}{%
11386 \string' \string'\string;
11387 \string'\glshex 00A0\string'\string;
11388 \string'\glshex 2000\string'\string;
11389 \string'\glshex 2001\string'\string;
11390 \string'\glshex 2002\string'\string;
11391 \string'\glshex 2003\string'\string;
11392 \string'\glshex 2004\string'\string;
11393 \string'\glshex 2005\string'\string;
11394 \string'\glshex 2006\string'\string;
11395 \string'\glshex 2007\string'\string;

```

```

11396 \string'\glshex 2008\string'\string;
11397 \string'\glshex 2009\string'\string;
11398 \string'\glshex 200A\string'\string;
11399 \string'\glshex 3000\string'
11400 }

```

nprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```

11401 \newcommand*{\glxtrnonprintablerules}{%
11402 \string'\glshex FEFF\string'\string;
11403 \string'\glshex 000A\string'\string;
11404 \string'\glshex 0009\string'\string;
11405 \string'\glshex 000C\string'\string;
11406 \string'\glshex 000B\string'
11407 }

```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```

11408 \newcommand*{\glxtrcombiningdiacriticrules}{%
11409 \glxtrcombiningdiacriticIrules\string;
11410 \glxtrcombiningdiacriticIIrules\string;
11411 \glxtrcombiningdiacriticIIIrules\string;
11412 \glxtrcombiningdiacriticIVrules
11413 }

```

diacriticIrules First set of combining diacritic marks.

```

11414 \newcommand*{\glxtrcombiningdiacriticIrules}{%
11415 \glshex 0301\string;% combining acute
11416 \glshex 0300\string;% combining grave
11417 \glshex 0306\string;% combining breve
11418 \glshex 0302\string;% combining circumflex
11419 \glshex 030C\string;% combining caron
11420 \glshex 030A\string;% combining ring
11421 \glshex 030D\string;% combining vertical line above
11422 \glshex 0308\string;% combining diaeresis
11423 \glshex 030B\string;% combining double acute
11424 \glshex 0303\string;% combining tilde
11425 \glshex 0307\string;% combining dot above
11426 \glshex 0304\string;% combining macron
11427 }

```

diacriticIIrules Second set of combining diacritic marks.

```

11428 \newcommand*{\glxtrcombiningdiacriticIIrules}{%
11429 \glshex 0337\string;% combining short solidus overlay
11430 \glshex 0327\string;% combining cedilla
11431 \glshex 0328\string;% combining ogonek
11432 \glshex 0323\string;% combining dot below
11433 \glshex 0332\string;% combining low line
11434 \glshex 0305\string;% combining overline
11435 \glshex 0309\string;% combining hook above
11436 \glshex 030E\string;% combining double vertical line above

```

11437 \glshex 030F\string;% combining double grave accent
 11438 \glshex 0310\string;% combining candrabindu
 11439 \glshex 0311\string;% combining inverted breve
 11440 \glshex 0312\string;% combining turned comma above
 11441 \glshex 0313\string;% combining comma above
 11442 \glshex 0314\string;% combining reversed comma above
 11443 \glshex 0315\string;% combining comma above right
 11444 \glshex 0316\string;% combining grave accent below
 11445 \glshex 0317% combining acute accent below
 11446 }

acriticIIIrules Third set of combining diacritic marks.

11447 \newcommand*{\glxtrcombiningdiacriticIIIrules}{%
 11448 \glshex 0318\string;% combining left tack below
 11449 \glshex 0319\string;% combining right tack below
 11450 \glshex 031A\string;% combining left angle above
 11451 \glshex 031B\string;% combining horn
 11452 \glshex 031C\string;% combining left half ring below
 11453 \glshex 031D\string;% combining up tack below
 11454 \glshex 031E\string;% combining down tack below
 11455 \glshex 031F\string;% combining plus sign below
 11456 \glshex 0320\string;% combining minus sign below
 11457 \glshex 0321\string;% combining palatalized hook below
 11458 \glshex 0322\string;% combining retroflex hook below
 11459 \glshex 0324\string;% combining diaeresis below
 11460 \glshex 0325\string;% combining ring below
 11461 \glshex 0326\string;% combining comma below
 11462 \glshex 0329\string;% combining vertical line below
 11463 \glshex 032A\string;% combining bridge below
 11464 \glshex 032B\string;% combining inverted double arch below
 11465 \glshex 032C\string;% combining caron below
 11466 \glshex 032D\string;% combining circumflex accent below
 11467 \glshex 032E\string;% combining breve below
 11468 \glshex 032F\string;% combining inverted breve below
 11469 \glshex 0330\string;% combining tilde below
 11470 \glshex 0331\string;% combining macron below
 11471 \glshex 0333\string;% combining double low line
 11472 \glshex 0334\string;% combining tilde overlay
 11473 \glshex 0335\string;% combining short stroke overlay
 11474 \glshex 0336\string;% combining long stroke overlay
 11475 \glshex 0338\string;% combining long solidus overlay
 11476 \glshex 0339\string;% combining combining right half ring below
 11477 \glshex 033A\string;% combining inverted bridge below
 11478 \glshex 033B\string;% combining square below
 11479 \glshex 033C\string;% combining seagull below
 11480 \glshex 033D\string;% combining x above
 11481 \glshex 033E\string;% combining vertical tilde
 11482 \glshex 033F\string;% combining double overline
 11483 \glshex 0342\string;% combining Greek perispomeni

```

11484 \glshex 0344\string;% combining Greek dialytika tonos
11485 \glshex 0345\string;% combining Greek ypogegrammeni
11486 \glshex 0360\string;% combining double tilde
11487 \glshex 0361\string;% combining double inverted breve
11488 \glshex 0483\string;% combining Cyrillic titlo
11489 \glshex 0484\string;% combining Cyrillic palatalization
11490 \glshex 0485\string;% combining Cyrillic dasia pneumata
11491 \glshex 0486% combining Cyrillic psili pneumata
11492 }

```

iacriticIVrules Fourth set of combining diacritic marks.

```

11493 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
11494 \glshex 20D0\string;% combining left harpoon above
11495 \glshex 20D1\string;% combining right harpoon above
11496 \glshex 20D2\string;% combining long vertical line overlay
11497 \glshex 20D3\string;% combining short vertical line overlay
11498 \glshex 20D4\string;% combining anticlockwise arrow above
11499 \glshex 20D5\string;% combining clockwise arrow above
11500 \glshex 20D6\string;% combining left arrow above
11501 \glshex 20D7\string;% combining right arrow above
11502 \glshex 20D8\string;% combining ring overlay
11503 \glshex 20D9\string;% combining clockwise ring overlay
11504 \glshex 20DA\string;% combining anticlockwise ring overlay
11505 \glshex 20DB\string;% combining three dots above
11506 \glshex 20DC\string;% combining four dots above
11507 \glshex 20DD\string;% combining enclosing circle
11508 \glshex 20DE\string;% combining enclosing square
11509 \glshex 20DF\string;% combining enclosing diamond
11510 \glshex 20E0\string;% combining enclosing circle backslash
11511 \glshex 20E1% combining left right arrow above
11512 }

```

sxtrhyphenrules Hyphens.

```

11513 \newcommand*{\glxtrhyphenrules}{%
11514 \string'\string-\string'\string;% ASCII hyphen
11515 \glshex 00AD\string;% soft hyphen
11516 \glshex 2010\string;% hyphen
11517 \glshex 2011\string;% non-breaking hyphen
11518 \glshex 2012\string;% figure dash
11519 \glshex 2013\string;% en dash
11520 \glshex 2014\string;% em dash
11521 \glshex 2015\string;% horizontal bar
11522 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
11523 }

```

eneralpuncrules General punctuation.

```

11524 \newcommand*{\glxtrgeneralpuncrules}{%
11525 \glxtrgeneralpuncrules
11526 \string<\glxtrcurrencyrules

```

```

11527 \string<\glxtrgeneralpuncIrules
11528 }

```

neralpuncIrules First set of general punctuation.

```

11529 \newcommand*{\glxtrgeneralpuncIrules}{%
11530 \string'\glshex 005F\string'% underscore
11531 \string<\glshex 00AF\string'% macron
11532 \string<\string'\glshex 002C\string'% comma
11533 \string<\string'\glshex 003B\string'% semi-colon
11534 \string<\string'\glshex 003A\string'% colon
11535 \string<\string'\glshex 0021\string'% exclamation mark
11536 \string<\glshex 00A1\string'% inverted exclamation mark
11537 \string<\string'\glshex 003F\string'% question mark
11538 \string<\glshex 00BF\string'% inverted question mark
11539 \string<\string'\glshex 002F\string'% solidus
11540 \string<\string'\glshex 002E\string'% full stop
11541 \string<\glshex 00B4\string'% acute accent
11542 \string<\string'\glshex 0060\string'% grave accent
11543 \string<\string'\glshex 005E\string'% circumflex accent
11544 \string<\glshex 00A8\string'% diaeresis
11545 \string<\string'\glshex 007E\string'% tilde
11546 \string<\glshex 00B7\string'% middle dot
11547 \string<\glshex 00B8\string'% cedilla
11548 \string<\string'\glshex 0027\string'% straight apostrophe
11549 \string<\string'\glshex 0022\string'% straight double quote
11550 \string<\glshex 00AB\string'% left guillemet
11551 \string<\glshex 00BB\string'% right guillemet
11552 \string<\string'\glshex 0028\string'% left parenthesis
11553 \string=<\glshex 0029\string'% right parenthesis
11554 \string=<\glshex 0028\string'% left parenthesis
11555 \string=<\glshex 0029\string'% right parenthesis
11556 \string<\string'\glshex 005B\string'% left square bracket
11557 \string<\string'\glshex 005D\string'% right square bracket
11558 \string<\string'\glshex 007B\string'% left curly bracket
11559 \string<\string'\glshex 007D\string'% right curly bracket
11560 \string<\glshex 00A7\string'% section sign
11561 \string<\glshex 00B6\string'% pilcrow sign
11562 \string<\glshex 00A9\string'% copyright sign
11563 \string<\glshex 00AE\string'% registered sign
11564 \string<\string'\glshex 0040\string'% at sign
11565 }

```

trcurrencyrules General punctuation.

```

11566 \newcommand*{\glxtrcurrencyrules}{%
11567 \glshex 00A4\string'% currency sign
11568 \string<\glshex 0E3F\string'% Thai currency symbol baht
11569 \string<\glshex 00A2\string'% cent sign
11570 \string<\glshex 20A1\string'% colon sign
11571 \string<\glshex 20A2\string'% cruzeiro sign

```

```

11572 \string<\string'\glshex 0024\string'% dollar sign
11573 \string<\glshex 20AB% dong sign
11574 \string<\glshex 20AC% euro sign
11575 \string<\glshex 20A3% French franc sign
11576 \string<\glshex 20A4% lira sign
11577 \string<\glshex 20A5% mill sign
11578 \string<\glshex 20A6% naira sign
11579 \string<\glshex 20A7% peseta sign
11580 \string<\glshex 00A3% pound sign
11581 \string<\glshex 20A8% rupee sign
11582 \string<\glshex 20AA% new sheqel sign
11583 \string<\glshex 20A9% won sign
11584 \string<\glshex 00A5% yen sign
11585 }

```

eralpuncIIrules Second set of general punctuation.

```

11586 \newcommand*{\glxtrgeneralpuncIIrules}{%
11587 \string'\glshex 002A\string'% asterisk
11588 \string<\string'\glshex 005C\string'% backslash
11589 \string<\string'\glshex 0026\string'% ampersand
11590 \string<\string'\glshex 0023\string'% hash sign
11591 \string<\string'\glshex 0025\string'% percent sign
11592 \string<\string'\glshex 002B\string'% plus sign
11593 \string=<\glshex 207A\string=<\glshex 208A% super/subscript plus sign
11594 \string<\glshex 00B1% plus-minus sign
11595 \string<\glshex 00F7% division sign
11596 \string<\glshex 00D7% multiplication sign
11597 \string<\string'\glshex 003C\string'% less-than sign
11598 \string<\string'\glshex 003D\string'% equals sign
11599 \string<\string'\glshex 003E\string'% greater-than sign
11600 \string<\glshex 00AC% not sign
11601 \string<\string'\glshex 007C\string'% vertical bar (pipe)
11602 \string<\glshex 00A6% broken bar
11603 \string<\glshex 00B0% degree sign
11604 \string<\glshex 00B5% micron sign
11605 }

```

eralLatinIrules Basic Latin alphabet.

```

11606 \newcommand*{\glxtrGeneralLatinIrules}{%
11607 \glxtrLatinA
11608 \string<b,B%
11609 \string<c,C%
11610 \string<d,D%
11611 \string<\glxtrLatinE
11612 \string<f,F%
11613 \string<g,G%
11614 \string<\glxtrLatinH
11615 \string<\glxtrLatinI
11616 \string<j,J%

```

```

11617 \string<\glxtrLatinK
11618 \string<\glxtrLatinL
11619 \string<\glxtrLatinM
11620 \string<\glxtrLatinN
11621 \string<\glxtrLatinO
11622 \string<\glxtrLatinP
11623 \string<q,Q%
11624 \string<r,R%
11625 \string<\glxtrLatinS
11626 \string<\glxtrLatinT
11627 \string<u,U%
11628 \string<v,V%
11629 \string<w,W%
11630 \string<\glxtrLatinX
11631 \string<y,Y%
11632 \string<z,Z
11633 }

```

raLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

11634 \newcommand*{\glxtrGeneralLatinIIrules}{%
11635 \glxtrLatinA
11636 \string<b,B%
11637 \string<c,C%
11638 \string<d,D%
11639 \string<\glxtrLatinEth
11640 \string<\glxtrLatinE
11641 \string<f,F%
11642 \string<g,G%
11643 \string<\glxtrLatinH
11644 \string<\glxtrLatinI
11645 \string<j,J%
11646 \string<\glxtrLatinK
11647 \string<\glxtrLatinL
11648 \string<\glxtrLatinM
11649 \string<\glxtrLatinN
11650 \string<\glxtrLatinO
11651 \string<\glxtrLatinP
11652 \string<q,Q%
11653 \string<r,R%
11654 \string<\glxtrLatinS
11655 \string& SS \string, \glxtrLatinEszettSs
11656 \string<\glxtrLatinT
11657 \string<u,U%
11658 \string<v,V%
11659 \string<w,W%
11660 \string<\glxtrLatinX
11661 \string<y,Y%
11662 \string<z,Z%
11663 }

```

alLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```
11664 \newcommand*{\glxtrGeneralLatinIIIrules}{%
11665 \glxtrLatinA
11666 \string<b,B%
11667 \string<c,C%
11668 \string<d,D%
11669 \string<\glxtrLatinEth
11670 \string<\glxtrLatinE
11671 \string<f,F%
11672 \string<g,G%
11673 \string<\glxtrLatinH
11674 \string<\glxtrLatinI
11675 \string<j,J%
11676 \string<\glxtrLatinK
11677 \string<\glxtrLatinL
11678 \string<\glxtrLatinM
11679 \string<\glxtrLatinN
11680 \string<\glxtrLatinO
11681 \string<\glxtrLatinP
11682 \string<q,Q%
11683 \string<r,R%
11684 \string<\glxtrLatinS
11685 \string& SZ, \glxtrLatinEszettSz
11686 \string<\glxtrLatinT
11687 \string<u,U%
11688 \string<v,V%
11689 \string<w,W%
11690 \string<\glxtrLatinX
11691 \string<y,Y%
11692 \string<z,Z%
11693 }
```

alLatinIVrules 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).

```
11694 \newcommand*{\glxtrGeneralLatinIVrules}{%
11695 \glxtrLatinA
11696 \string& AE , \glxtrLatinAELigature
11697 \string<b,B%
11698 \string<c,C%
11699 \string<d,D%
11700 \string<\glxtrLatinEth
11701 \string<\glxtrLatinE
11702 \string<f,F%
11703 \string<g,G%
11704 \string<\glxtrLatinH
11705 \string<\glxtrLatinI
11706 \string<j,J%
11707 \string<\glxtrLatinK
11708 \string<\glxtrLatinL
```

```

11709 \string<\glxtrLatinM
11710 \string<\glxtrLatinN
11711 \string<\glxtrLatinO
11712 \string& OE , \glxtrLatinOELigature
11713 \string<\glxtrLatinP
11714 \string<q,Q%
11715 \string<r,R%
11716 \string<\glxtrLatinS
11717 \string& SS , \glxtrLatinEszettSs
11718 \string<\glxtrLatinT
11719 \string& th =\glshex 00DE
11720 \string& TH =\glshex 00FE
11721 \string<u,U%
11722 \string<v,V%
11723 \string<w,W%
11724 \string<\glxtrLatinX
11725 \string<y,Y%
11726 \string<z,Z%
11727 }

```

eralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

11728 \newcommand*{\glxtrGeneralLatinVrules}{%
11729 \glxtrLatinA
11730 \string<b,B%
11731 \string<c,C%
11732 \string<d,D%
11733 \string<\glxtrLatinEth
11734 \string<\glxtrLatinE
11735 \string<f,F%
11736 \string<g,G%
11737 \string<\glxtrLatinH
11738 \string<\glxtrLatinI
11739 \string<j,J%
11740 \string<\glxtrLatinK
11741 \string<\glxtrLatinL
11742 \string<\glxtrLatinM
11743 \string<\glxtrLatinN
11744 \string<\glxtrLatinO
11745 \string<\glxtrLatinP
11746 \string<q,Q%
11747 \string<r,R%
11748 \string<\glxtrLatinS
11749 \string& SS , \glxtrLatinEszettSs
11750 \string<\glxtrLatinT
11751 \string& th =\glshex 00DE
11752 \string& TH =\glshex 00FE
11753 \string<u,U%
11754 \string<v,V%
11755 \string<w,W%

```

```

11756 \string<\glxtrLatinX
11757 \string<y,Y%
11758 \string<z,Z%
11759 }

```

raLatinVIrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

11760 \newcommand*{\glxtrGeneralLatinVIrules}{%
11761 \glxtrLatinA
11762 \string<b,B%
11763 \string<c,C%
11764 \string<d,D%
11765 \string<\glxtrLatinEth
11766 \string<\glxtrLatinE
11767 \string<f,F%
11768 \string<g,G%
11769 \string<\glxtrLatinH
11770 \string<\glxtrLatinI
11771 \string<j,J%
11772 \string<\glxtrLatinK
11773 \string<\glxtrLatinL
11774 \string<\glxtrLatinM
11775 \string<\glxtrLatinN
11776 \string<\glxtrLatinO
11777 \string<\glxtrLatinP
11778 \string<q,Q%
11779 \string<r,R%
11780 \string<\glxtrLatinS
11781 \string& SZ , \glxtrLatinEszettSz
11782 \string<\glxtrLatinT
11783 \string& th =\glshex 00DE
11784 \string& TH =\glshex 00FE
11785 \string<u,U%
11786 \string<v,V%
11787 \string<w,W%
11788 \string<\glxtrLatinX
11789 \string<y,Y%
11790 \string<z,Z%
11791 }

```

alLatinVIIrules General Latin alphabet (Æ between A and B, eth between D and E, insular G as G, Œ between O and P, long S equivalent to S, Þ between T and U and wynn as W).

```

11792 \newcommand*{\glxtrGeneralLatinVIIrules}{%
11793 \glxtrLatinA
11794 \string<\glxtrLatinAELigature
11795 \string<b,B%
11796 \string<c,C%
11797 \string<d,D%
11798 \string<\glxtrLatinEth
11799 \string<\glxtrLatinE

```

```

11800 \string<f,F%
11801 \string<\glxtrLatinInsularG
11802 \string<\glxtrLatinH
11803 \string<\glxtrLatinI
11804 \string<j,J%
11805 \string<\glxtrLatinK
11806 \string<\glxtrLatinL
11807 \string<\glxtrLatinM
11808 \string<\glxtrLatinN
11809 \string<\glxtrLatinO
11810 \string<\glxtrLatinOELigature
11811 \string<\glxtrLatinP
11812 \string<q,Q%
11813 \string<r,R%
11814 \string<\glshex 017F=\glxtrLatinS % s and long s
11815 \string<\glxtrLatinT
11816 \string<\glxtrLatinThorn
11817 \string<u,U%
11818 \string<v,V%
11819 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
11820 \string<\glxtrLatinX
11821 \string<y,Y%
11822 \string<z,Z%
11823 }

```

11LatinVIIIrules General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth treated as D, Ø treated as O, Ł treated as L).

```

11824 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
11825 \glxtrLatinA
11826 \string& AE , \glxtrLatinAELigature
11827 \string<b,B%
11828 \string<c,C%
11829 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
11830 \string<\glxtrLatinE
11831 \string<f,F%
11832 \string<g,G%
11833 \string<\glxtrLatinH
11834 \string<\glxtrLatinI
11835 \string<j,J%
11836 \string<\glxtrLatinK
11837 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
11838 \string<\glxtrLatinM
11839 \string<\glxtrLatinN
11840 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
11841 \string& OE , \glxtrLatinOELigature
11842 \string<\glxtrLatinP
11843 \string<q,Q%
11844 \string<r,R%
11845 \string<\glxtrLatinS

```

```

11846 \string& SS , \glxtrLatinEszettSs
11847 \string<\glxtrLatinT
11848 \string& th =\glshex 00DE
11849 \string& TH =\glshex 00FE
11850 \string<u,U%
11851 \string<v,V%
11852 \string<w,W%
11853 \string<\glxtrLatinX
11854 \string<y,Y%
11855 \string<z,Z%
11856 }

```

\glxtrLatinA

```

11857 \newcommand*{\glxtrLatinA}{%
11858   a\string=\glshex 00AA\string=\glshex 2090,A
11859 }

```

\glxtrLatinE

```

11860 \newcommand*{\glxtrLatinE}{%
11861   e\string=\glshex 2091,E
11862 }

```

\glxtrLatinH

```

11863 \newcommand*{\glxtrLatinH}{%
11864   h\string=\glshex 2095,H
11865 }

```

\glxtrLatinI

```

11866 \newcommand*{\glxtrLatinI}{%
11867   i\string=\glshex 2071,I
11868 }

```

\glxtrLatinK

```

11869 \newcommand*{\glxtrLatinK}{%
11870   k\string=\glshex 2096,K
11871 }

```

\glxtrLatinL

```

11872 \newcommand*{\glxtrLatinL}{%
11873   l\string=\glshex 2097,L
11874 }

```

\glxtrLatinM

```

11875 \newcommand*{\glxtrLatinM}{%
11876   m\string=\glshex 2098,M
11877 }

```

\glsxtrLatinN

```
11878 \newcommand*{\glsxtrLatinN}{%
11879   n\string=\glshex 207F\string=\glshex 2099,N
11880 }
```

\glsxtrLatinO

```
11881 \newcommand*{\glsxtrLatinO}{%
11882   o\string=\glshex 00BA\string=\glshex 2092,O
11883 }
```

\glsxtrLatinP

```
11884 \newcommand*{\glsxtrLatinP}{%
11885   p\string=\glshex 209A,P
11886 }
```

\glsxtrLatinS

```
11887 \newcommand*{\glsxtrLatinS}{%
11888   s\string=\glshex 209B,S
11889 }
```

\glsxtrLatinT

```
11890 \newcommand*{\glsxtrLatinT}{%
11891   t\string=\glshex 209C,T
11892 }
```

\glsxtrLatinX

```
11893 \newcommand*{\glsxtrLatinX}{%
11894   x\string=\glshex 2093,X
11895 }
```

lsxtrLatinSchwa Latin schwa (lower case, subscript and upper case).

```
11896 \newcommand*{\glsxtrLatinSchwa}{%
11897   \glshex 0259\string=\glshex 2094,\glshex 018F
11898 }
```

trLatinEszettSs

```
11899 \newcommand*{\glsxtrLatinEszettSs}{%
11900   \glshex 00DF% eszett
11901   \string=\glshex 017Fs % long S s
11902 }
```

trLatinEszettSz

```
11903 \newcommand*{\glsxtrLatinEszettSz}{%
11904   \glshex 00DF% eszett
11905   \string= \glshex 017Fz % long S z
11906 }
```

\glxtrLatinEth

```
11907 \newcommand*{\glxtrLatinEth}{%
11908   \glshex 00F0,\glshex 00D0% eth
11909 }
```

lsxtrLatinThorn

```
11910 \newcommand*{\glxtrLatinThorn}{%
11911   \glshex 00FE,\glshex 00DE% thorn
11912 }
```

LatinAELigature

```
11913 \newcommand*{\glxtrLatinAELigature}{%
11914   \glshex 00E6,\glshex 00C6% AE-ligature
11915 }
```

LatinOELigature

```
11916 \newcommand*{\glxtrLatinOELigature}{%
11917   \glshex 0153,\glshex 0152% OE-ligature
11918 }
```

\glxtrLatinAA

```
11919 \newcommand*{\glxtrLatinAA}{%
11920   \glshex 00E5=a\glshex 030A,% \aa
11921   \glshex 00C5=A\glshex 030A% \AA
11922 }
```

glxtrLatinWynn

```
11923 \newcommand*{\glxtrLatinWynn}{%
11924   \glshex 01BF,\glshex 01F7% wynn
11925 }
```

trLatinInsularG

```
11926 \newcommand*{\glxtrLatinInsularG}{%
11927   \glshex 1D79,\glshex A77D% insular G
11928   \string; g, G
11929 }
```

sxtrLatinOslash

```
11930 \newcommand*{\glxtrLatinOslash}{%
11931   \glshex 00F8,\glshex 00D8% \o, \O
11932 }
```

sxtrLatinLslash

```
11933 \newcommand*{\glxtrLatinLslash}{%
11934   \glshex 0142,\glshex 0141% \l, \L
11935 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
11936 \newcommand*{\glxtrMathUpGreekIrules}{%
11937 \glxtrUpAlpha
11938 \string<\glxtrUpBeta
11939 \string<\glxtrUpGamma
11940 \string<\glxtrUpDelta
11941 \string<\glxtrUpEpsilon
11942 \string<\glxtrUpDigamma
11943 \string<\glxtrUpZeta
11944 \string<\glxtrUpEta
11945 \string<\glxtrUpTheta
11946 \string<\glxtrUpIota
11947 \string<\glxtrUpKappa
11948 \string<\glxtrUpLambda
11949 \string<\glxtrUpMu
11950 \string<\glxtrUpNu
11951 \string<\glxtrUpXi
11952 \string<\glxtrUpOmicron
11953 \string<\glxtrUpPi
11954 \string<\glxtrUpRho
11955 \string<\glxtrUpSigma
11956 \string<\glxtrUpTau
11957 \string<\glxtrUpUpsilon
11958 \string<\glxtrUpPhi
11959 \string<\glxtrUpChi
11960 \string<\glxtrUpPsi
11961 \string<\glxtrUpOmega
11962 }
```

hUpGreekIIrules Doesn't include digamma.

```
11963 \newcommand*{\glxtrMathUpGreekIIrules}{%
11964 \glxtrUpAlpha
11965 \string<\glxtrUpBeta
11966 \string<\glxtrUpGamma
11967 \string<\glxtrUpDelta
11968 \string<\glxtrUpEpsilon
11969 \string<\glxtrUpZeta
11970 \string<\glxtrUpEta
11971 \string<\glxtrUpTheta
11972 \string<\glxtrUpIota
11973 \string<\glxtrUpKappa
11974 \string<\glxtrUpLambda
11975 \string<\glxtrUpMu
11976 \string<\glxtrUpNu
11977 \string<\glxtrUpXi
11978 \string<\glxtrUpOmicron
11979 \string<\glxtrUpPi
11980 \string<\glxtrUpRho
11981 \string<\glxtrUpSigma
```

```

11982 \string<\glxtrUpTau
11983 \string<\glxtrUpUpsilon
11984 \string<\glxtrUpPhi
11985 \string<\glxtrUpChi
11986 \string<\glxtrUpPsi
11987 \string<\glxtrUpOmega
11988 }

```

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.

```

11989 \newcommand*{\glxtrMathItalicGreekIrules}{%
11990 \glxtrMathItalicAlpha
11991 \string<\glxtrMathItalicBeta
11992 \string<\glxtrMathItalicGamma
11993 \string<\glxtrMathItalicDelta
11994 \string<\glxtrMathItalicEpsilon
11995 \string<\glxtrUpDigamma
11996 \string<\glxtrMathItalicZeta
11997 \string<\glxtrMathItalicEta
11998 \string<\glxtrMathItalicTheta
11999 \string<\glxtrMathItalicIota
12000 \string<\glxtrMathItalicKappa
12001 \string<\glxtrMathItalicLambda
12002 \string<\glxtrMathItalicMu
12003 \string<\glxtrMathItalicNu
12004 \string<\glxtrMathItalicXi
12005 \string<\glxtrMathItalicOmicron
12006 \string<\glxtrMathItalicPi
12007 \string<\glxtrMathItalicRho
12008 \string<\glxtrMathItalicSigma
12009 \string<\glxtrMathItalicTau
12010 \string<\glxtrMathItalicUpsilon
12011 \string<\glxtrMathItalicPhi
12012 \string<\glxtrMathItalicChi
12013 \string<\glxtrMathItalicPsi
12014 \string<\glxtrMathItalicOmega
12015 }

```

alicGreekIIrules Doesn't include digamma.

```

12016 \newcommand*{\glxtrMathItalicGreekIIrules}{%
12017 \glxtrMathItalicAlpha
12018 \string<\glxtrMathItalicBeta
12019 \string<\glxtrMathItalicGamma
12020 \string<\glxtrMathItalicDelta
12021 \string<\glxtrMathItalicEpsilon
12022 \string<\glxtrMathItalicZeta
12023 \string<\glxtrMathItalicEta
12024 \string<\glxtrMathItalicTheta
12025 \string<\glxtrMathItalicIota

```

```

12026 \string<\glxtrMathItalicKappa
12027 \string<\glxtrMathItalicLambda
12028 \string<\glxtrMathItalicMu
12029 \string<\glxtrMathItalicNu
12030 \string<\glxtrMathItalicXi
12031 \string<\glxtrMathItalicOmicron
12032 \string<\glxtrMathItalicPi
12033 \string<\glxtrMathItalicRho
12034 \string<\glxtrMathItalicSigma
12035 \string<\glxtrMathItalicTau
12036 \string<\glxtrMathItalicUpsilon
12037 \string<\glxtrMathItalicPhi
12038 \string<\glxtrMathItalicChi
12039 \string<\glxtrMathItalicPsi
12040 \string<\glxtrMathItalicOmega
12041 }

```

UpperGreekIrules Upper case only (includes upright digamma).

```

12042 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
12043 \glshex 1D6E2% upper case alpha (maths italic)
12044 \string<\glshex 1D6E3% upper case beta (maths italic)
12045 \string<\glshex 1D6E4% upper case gamma (maths italic)
12046 \string<\glshex 1D6E5% upper case delta (maths italic)
12047 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12048 \string<\glshex 03DC% upper case digamma
12049 \string<\glshex 1D6E7% upper case zeta (maths italic)
12050 \string<\glshex 1D6E8% upper case eta (maths italic)
12051 \string<\glshex 1D6E9% upper case theta (maths italic)
12052 \string<\glshex 1D6F3% upper case theta variant (maths italic)
12053 \string<\glshex 1D6EA% upper case iota (maths italic)
12054 \string<\glshex 1D6EB% upper case kappa (maths italic)
12055 \string<\glshex 1D6EC% upper case lambda (maths italic)
12056 \string<\glshex 1D6ED% upper case mu (maths italic)
12057 \string<\glshex 1D6EE% upper case nu (maths italic)
12058 \string<\glshex 1D6EF% upper case xi (maths italic)
12059 \string<\glshex 1D6F0% upper case omicron (maths italic)
12060 \string<\glshex 1D6F1% upper case pi (maths italic)
12061 \string<\glshex 1D6F2% upper case rho (maths italic)
12062 \string<\glshex 1D6F4% upper case sigma (maths italic)
12063 \string<\glshex 1D6F5% upper case tau (maths italic)
12064 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12065 \string<\glshex 1D6F7% upper case phi (maths italic)
12066 \string<\glshex 1D6F8% upper case chi (maths italic)
12067 \string<\glshex 1D6F9% upper case psi (maths italic)
12068 \string<\glshex 1D6FA% upper case omega (maths italic)
12069 }

```

UpperGreekIIrules Upper case only (doesn't include upright digamma).

```

12070 \newcommand*{\glxtrMathItalicUpperGreekIIrules}{%

```

```

12071 \glshex 1D6E2% upper case alpha (maths italic)
12072 \string<\glshex 1D6E3% upper case beta (maths italic)
12073 \string<\glshex 1D6E4% upper case gamma (maths italic)
12074 \string<\glshex 1D6E5% upper case delta (maths italic)
12075 \string<\glshex 1D6E6% upper case epsilon (maths italic)
12076 \string<\glshex 1D6E7% upper case zeta (maths italic)
12077 \string<\glshex 1D6E8% upper case eta (maths italic)
12078 \string<\glshex 1D6E9% upper case theta (maths italic)
12079 \string=\glshex 1D6F3% upper case theta variant (maths italic)
12080 \string<\glshex 1D6EA% upper case iota (maths italic)
12081 \string<\glshex 1D6EB% upper case kappa (maths italic)
12082 \string<\glshex 1D6EC% upper case lambda (maths italic)
12083 \string<\glshex 1D6ED% upper case mu (maths italic)
12084 \string<\glshex 1D6EE% upper case nu (maths italic)
12085 \string<\glshex 1D6EF% upper case xi (maths italic)
12086 \string<\glshex 1D6F0% upper case omicron (maths italic)
12087 \string<\glshex 1D6F1% upper case pi (maths italic)
12088 \string<\glshex 1D6F2% upper case rho (maths italic)
12089 \string<\glshex 1D6F4% upper case sigma (maths italic)
12090 \string<\glshex 1D6F5% upper case tau (maths italic)
12091 \string<\glshex 1D6F6% upper case upsilon (maths italic)
12092 \string<\glshex 1D6F7% upper case phi (maths italic)
12093 \string<\glshex 1D6F8% upper case chi (maths italic)
12094 \string<\glshex 1D6F9% upper case psi (maths italic)
12095 \string<\glshex 1D6FA% upper case omega (maths italic)
12096 }

```

LowerGreekIrules Lower case only (includes upright digamma).

```

12097 \newcommand*{\glxtrMathItalicLowerGreekIrules}{%
12098 \glshex 1D6FC% lower case alpha (maths italic)
12099 \string<\glshex 1D6FD% lower case beta (maths italic)
12100 \string<\glshex 1D6FE% lower case gamma (maths italic)
12101 \string<\glshex 1D6FF% lower case delta (maths italic)
12102 \string<\glshex 1D700% lower case epsilon (maths italic)
12103 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12104 \string<\glshex 03DD% lower case digamma
12105 \string<\glshex 1D701% lower case zeta (maths italic)
12106 \string<\glshex 1D702% lower case eta (maths italic)
12107 \string<\glshex 1D703% lower case theta (maths italic)
12108 \string=\glshex 1D717% lower case theta variant (maths italic)
12109 \string<\glshex 1D704% lower case iota (maths italic)
12110 \string<\glshex 1D705% lower case kappa (maths italic)
12111 \string=\glshex 1D718% lower case kappa variant (maths italic)
12112 \string<\glshex 1D706% lower case lambda (maths italic)
12113 \string<\glshex 1D707% lower case mu (maths italic)
12114 \string<\glshex 1D708% lower case nu (maths italic)
12115 \string<\glshex 1D709% lower case xi (maths italic)
12116 \string<\glshex 1D70A% lower case omicron (maths italic)
12117 \string<\glshex 1D70B% lower case pi (maths italic)

```

```

12118 \string=\glshex 1D71B% lower case pi variant (maths italic)
12119 \string<\glshex 1D70C% lower case rho (maths italic)
12120 \string=\glshex 1D71A% lower case rho variant (maths italic)
12121 \string<\glshex 1D70D% lower case final sigma (maths italic)
12122 \string=\glshex 1D70E% lower case sigma (maths italic)
12123 \string<\glshex 1D70F% lower case tau (maths italic)
12124 \string<\glshex 1D710% lower case upsilon (maths italic)
12125 \string<\glshex 1D711% lower case phi (maths italic)
12126 \string=\glshex 1D719% lower case phi variant (maths italic)
12127 \string<\glshex 1D712% lower case chi (maths italic)
12128 \string<\glshex 1D713% lower case psi (maths italic)
12129 \string<\glshex 1D714% lower case omega (maths italic)
12130 }

```

LowerGreeklIrules Lower case only (doesn't includes upright digamma).

```

12131 \newcommand*{\glxtrMathItalicLowerGreeklIrules}{%
12132 \glshex 1D6FC% lower case alpha (maths italic)
12133 \string<\glshex 1D6FD% lower case beta (maths italic)
12134 \string<\glshex 1D6FE% lower case gamma (maths italic)
12135 \string<\glshex 1D6FF% lower case delta (maths italic)
12136 \string<\glshex 1D700% lower case epsilon (maths italic)
12137 \string=\glshex 1D716% lower case epsilon variant (maths italic)
12138 \string<\glshex 1D701% lower case zeta (maths italic)
12139 \string<\glshex 1D702% lower case eta (maths italic)
12140 \string<\glshex 1D703% lower case theta (maths italic)
12141 \string=\glshex 1D717% lower case theta variant (maths italic)
12142 \string<\glshex 1D704% lower case iota (maths italic)
12143 \string<\glshex 1D705% lower case kappa (maths italic)
12144 \string=\glshex 1D718% lower case kappa variant (maths italic)
12145 \string<\glshex 1D706% lower case lambda (maths italic)
12146 \string<\glshex 1D707% lower case mu (maths italic)
12147 \string<\glshex 1D708% lower case nu (maths italic)
12148 \string<\glshex 1D709% lower case xi (maths italic)
12149 \string<\glshex 1D70A% lower case omicron (maths italic)
12150 \string<\glshex 1D70B% lower case pi (maths italic)
12151 \string=\glshex 1D71B% lower case pi variant (maths italic)
12152 \string<\glshex 1D70C% lower case rho (maths italic)
12153 \string=\glshex 1D71A% lower case rho variant (maths italic)
12154 \string<\glshex 1D70D% lower case final sigma (maths italic)
12155 \string=\glshex 1D70E% lower case sigma (maths italic)
12156 \string<\glshex 1D70F% lower case tau (maths italic)
12157 \string<\glshex 1D710% lower case upsilon (maths italic)
12158 \string<\glshex 1D711% lower case phi (maths italic)
12159 \string=\glshex 1D719% lower case phi variant (maths italic)
12160 \string<\glshex 1D712% lower case chi (maths italic)
12161 \string<\glshex 1D713% lower case psi (maths italic)
12162 \string<\glshex 1D714% lower case omega (maths italic)
12163 }

```

MathGreekIrules Includes both upright and italic with digamma between epsilon and zeta.

```

12164 \newcommand*{\glxtrMathGreekIrules}{%
12165   \glxtrMathItalicAlpha
12166   \string;\glxtrUpAlpha
12167   \string<\glxtrMathItalicBeta
12168   \string;\glxtrUpBeta
12169   \string<\glxtrMathItalicGamma
12170   \string;\glxtrUpGamma
12171   \string<\glxtrMathItalicDelta
12172   \string;\glxtrUpDelta
12173   \string<\glxtrMathItalicEpsilon
12174   \string;\glxtrUpEpsilon
12175   \string<\glxtrUpDigamma
12176   \string<\glxtrMathItalicZeta
12177   \string;\glxtrUpZeta
12178   \string<\glxtrMathItalicEta
12179   \string;\glxtrUpEta
12180   \string<\glxtrMathItalicTheta
12181   \string;\glxtrUpTheta
12182   \string<\glxtrMathItalicIota
12183   \string;\glxtrUpIota
12184   \string<\glxtrMathItalicKappa
12185   \string;\glxtrUpKappa
12186   \string<\glxtrMathItalicLambda
12187   \string;\glxtrUpLambda
12188   \string<\glxtrMathItalicMu
12189   \string;\glxtrUpMu
12190   \string<\glxtrMathItalicNu
12191   \string;\glxtrUpNu
12192   \string<\glxtrMathItalicXi
12193   \string;\glxtrUpXi
12194   \string<\glxtrMathItalicOmicron
12195   \string;\glxtrUpOmicron
12196   \string<\glxtrMathItalicPi
12197   \string;\glxtrUpPi
12198   \string<\glxtrMathItalicRho
12199   \string;\glxtrUpRho
12200   \string<\glxtrMathItalicSigma
12201   \string;\glxtrUpSigma
12202   \string<\glxtrMathItalicTau
12203   \string;\glxtrUpTau
12204   \string<\glxtrMathItalicUpsilon
12205   \string;\glxtrUpUpsilon
12206   \string<\glxtrMathItalicPhi
12207   \string;\glxtrUpPhi
12208   \string<\glxtrMathItalicChi
12209   \string;\glxtrUpChi
12210   \string<\glxtrMathItalicPsi
12211   \string;\glxtrUpPsi

```

```

12212 \string<\glxtrMathItalicOmega
12213 \string;\glxtrUpOmega
12214 }

```

athGreekIIrules Includes both upright and italic (digamma not included).

```

12215 \newcommand*{\glxtrMathGreekIIrules}{%
12216 \glxtrMathItalicAlpha
12217 \string;\glxtrUpAlpha
12218 \string<\glxtrMathItalicBeta
12219 \string;\glxtrUpBeta
12220 \string<\glxtrMathItalicGamma
12221 \string;\glxtrUpGamma
12222 \string<\glxtrMathItalicDelta
12223 \string;\glxtrUpDelta
12224 \string<\glxtrMathItalicEpsilon
12225 \string;\glxtrUpEpsilon
12226 \string<\glxtrMathItalicZeta
12227 \string;\glxtrUpZeta
12228 \string<\glxtrMathItalicEta
12229 \string;\glxtrUpEta
12230 \string<\glxtrMathItalicTheta
12231 \string;\glxtrUpTheta
12232 \string<\glxtrMathItalicIota
12233 \string;\glxtrUpIota
12234 \string<\glxtrMathItalicKappa
12235 \string;\glxtrUpKappa
12236 \string<\glxtrMathItalicLambda
12237 \string;\glxtrUpLambda
12238 \string<\glxtrMathItalicMu
12239 \string;\glxtrUpMu
12240 \string<\glxtrMathItalicNu
12241 \string;\glxtrUpNu
12242 \string<\glxtrMathItalicXi
12243 \string;\glxtrUpXi
12244 \string<\glxtrMathItalicOmicron
12245 \string;\glxtrUpOmicron
12246 \string<\glxtrMathItalicPi
12247 \string;\glxtrUpPi
12248 \string<\glxtrMathItalicRho
12249 \string;\glxtrUpRho
12250 \string<\glxtrMathItalicSigma
12251 \string;\glxtrUpSigma
12252 \string<\glxtrMathItalicTau
12253 \string;\glxtrUpTau
12254 \string<\glxtrMathItalicUpsilon
12255 \string;\glxtrUpUpsilon
12256 \string<\glxtrMathItalicPhi
12257 \string;\glxtrUpPhi
12258 \string<\glxtrMathItalicChi

```

```

12259 \string;\glxtrUpChi
12260 \string<\glxtrMathItalicPsi
12261 \string;\glxtrUpPsi
12262 \string<\glxtrMathItalicOmega
12263 \string;\glxtrUpOmega
12264 }

```

\glxtrUpAlpha

```

12265 \newcommand*{\glxtrUpAlpha}{%
12266 \glshex 03B1,% lower case alpha
12267 \glshex 0391% upper case alpha
12268 }

```

\glxtrUpBeta

```

12269 \newcommand*{\glxtrUpBeta}{%
12270 \glshex 03B2,% lower case beta
12271 \glshex 0392% upper case beta
12272 }

```

\glxtrUpGamma

```

12273 \newcommand*{\glxtrUpGamma}{%
12274 \glshex 03B3,% lower case gamma
12275 \glshex 0393% upper case gamma
12276 }

```

\glxtrUpDelta

```

12277 \newcommand*{\glxtrUpDelta}{%
12278 \glshex 03B4,% lower case delta
12279 \glshex 0394% upper case delta
12280 }

```

glxtrUpEpsilon

```

12281 \newcommand*{\glxtrUpEpsilon}{%
12282 \glshex 03B5% lower case epsilon
12283 \string=\glshex 03F5,% lower case epsilon variant
12284 \glshex 0395% upper case epsilon
12285 }

```

glxtrUpDigamma

```

12286 \newcommand*{\glxtrUpDigamma}{%
12287 \glshex 03DD,% lower case digamma
12288 \glshex 03DC% upper case digamma
12289 }

```

\glxtrUpZeta

```

12290 \newcommand*{\glxtrUpZeta}{%
12291 \glshex 03B6,% lower case zeta
12292 \glshex 0396% upper case zeta
12293 }

```

$\backslash\mathrm{glxtrUpEta}$

```
12294 \newcommand*{\glxtrUpEta}{%
12295   \glshex 03B7,% lower case eta
12296   \glshex 0397% upper case eta
12297 }
```

$\backslash\mathrm{glxtrUpTheta}$

```
12298 \newcommand*{\glxtrUpTheta}{%
12299   \glshex 03B8% lower case theta
12300   \string=\glshex 03D1,% lower case theta variant
12301   \glshex 0398% upper case theta
12302 }
```

$\backslash\mathrm{glxtrUpIota}$

```
12303 \newcommand*{\glxtrUpIota}{%
12304   \glshex 03B9,% lower case iota
12305   \glshex 0399% upper case iota
12306 }
```

$\backslash\mathrm{glxtrUpKappa}$

```
12307 \newcommand*{\glxtrUpKappa}{%
12308   \glshex 03BA% lower case kappa
12309   \string=\glshex 03F0,% lower case kappa variant
12310   \glshex 039A% upper case kappa
12311 }
```

$\backslash\mathrm{glxtrUpLambda}$

```
12312 \newcommand*{\glxtrUpLambda}{%
12313   \glshex 03BB,% lower lambda
12314   \glshex 039B% upper case lambda
12315 }
```

$\backslash\mathrm{glxtrUpMu}$

```
12316 \newcommand*{\glxtrUpMu}{%
12317   \glshex 03BC,% lower case mu
12318   \glshex 039C% upper case mu
12319 }
```

$\backslash\mathrm{glxtrUpNu}$

```
12320 \newcommand*{\glxtrUpNu}{%
12321   \glshex 03BD,% lower case nu
12322   \glshex 039D% upper case nu
12323 }
```

$\backslash\mathrm{glxtrUpXi}$

```
12324 \newcommand*{\glxtrUpXi}{%
12325   \glshex 03BE,% lower case xi
12326   \glshex 039E% upper case xi
12327 }
```

glsxtrUpOmicron

```
12328 \newcommand*{\glsxtrUpOmicron}{%  
12329 \glsheX 03BF,% lower case omicron  
12330 \glsheX 039F% upper case omicron  
12331 }
```

\glsxtrUpPi

```
12332 \newcommand*{\glsxtrUpPi}{%  
12333 \glsheX 03C0% lower case pi  
12334 \string=\glsheX 03D6,% lower case pi variant  
12335 \glsheX 03A0% upper case pi  
12336 }
```

\glsxtrUpRho

```
12337 \newcommand*{\glsxtrUpRho}{%  
12338 \glsheX 03C1% lower case rho  
12339 \string=\glsheX 03F1,% lower case rho variant  
12340 \glsheX 03A1% upper case rho  
12341 }
```

\glsxtrUpSigma

```
12342 \newcommand*{\glsxtrUpSigma}{%  
12343 \glsheX 03C2% lower case sigma  
12344 \string=\glsheX 03C3,% lower case sigma  
12345 \glsheX 03A3% upper case sigma  
12346 }
```

\glsxtrUpTau

```
12347 \newcommand*{\glsxtrUpTau}{%  
12348 \glsheX 03C4,% lower case tau  
12349 \glsheX 03A4% upper case tau  
12350 }
```

glsxtrUpUpsilon

```
12351 \newcommand*{\glsxtrUpUpsilon}{%  
12352 \glsheX 03C5,% lower case upsilon  
12353 \glsheX 03A5% upper case upsilon  
12354 }
```

\glsxtrUpPhi

```
12355 \newcommand*{\glsxtrUpPhi}{%  
12356 \glsheX 03C6% lower case phi  
12357 \string=\glsheX 03D5,% lower case phi variant  
12358 \glsheX 03A6% upper case phi  
12359 }
```

\glsxtrUpChi

```
12360 \newcommand*{\glsxtrUpChi}{%
```

```

12361 \glshex 03C7,% lower case chi
12362 \glshex 03A7% upper case chi
12363 }

```

\glsxtrUpPsi

```

12364 \newcommand*{\glsxtrUpPsi}{%
12365 \glshex 03C8,% lower case psi
12366 \glshex 03A8% upper case psi
12367 }

```

\glsxtrUpOmega

```

12368 \newcommand*{\glsxtrUpOmega}{%
12369 \glshex 03C9,% lower case omega
12370 \glshex 03A9% upper case omega
12371 }

```

MathItalicAlpha

```

12372 \newcommand*{\glsxtrMathItalicAlpha}{%
12373 \glshex 1D6FC,% lower case alpha (maths italic)
12374 \glshex 1D6E2% upper case alpha (maths italic)
12375 }

```

rMathItalicBeta

```

12376 \newcommand*{\glsxtrMathItalicBeta}{%
12377 \glshex 1D6FD,% lower case beta (maths italic)
12378 \glshex 1D6E3% upper case beta (maths italic)
12379 }

```

MathItalicGamma

```

12380 \newcommand*{\glsxtrMathItalicGamma}{%
12381 \glshex 1D6FE,% lower case gamma (maths italic)
12382 \glshex 1D6E4% upper case gamma (maths italic)
12383 }

```

MathItalicDelta

```

12384 \newcommand*{\glsxtrMathItalicDelta}{%
12385 \glshex 1D6FF,% lower case delta (maths italic)
12386 \glshex 1D6E5% upper case delta (maths italic)
12387 }

```

thItalicEpsilon

```

12388 \newcommand*{\glsxtrMathItalicEpsilon}{%
12389 \glshex 1D700% lower case epsilon (maths italic)
12390 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
12391 \glshex 1D6E6% upper case epsilon (maths italic)
12392 }

```

rMathItalicZeta

```
12393 \newcommand*{\glxtrMathItalicZeta}{%
12394   \glshex 1D701,% lower case zeta (maths italic)
12395   \glshex 1D6E7% upper case zeta (maths italic)
12396 }
```

trMathItalicEta

```
12397 \newcommand*{\glxtrMathItalicEta}{%
12398   \glshex 1D702,% lower case eta (maths italic)
12399   \glshex 1D6E8% upper case eta (maths italic)
12400 }
```

MathItalicTheta

```
12401 \newcommand*{\glxtrMathItalicTheta}{%
12402   \glshex 1D703% lower case theta (maths italic)
12403   \string=\glshex 1D717,% lower case theta variant (maths italic)
12404   \glshex 1D6E9% upper case theta (maths italic)
12405   \string=\glshex 1D6F3% upper case theta variant (maths italic)
12406 }
```

rMathItalicIota

```
12407 \newcommand*{\glxtrMathItalicIota}{%
12408   \glshex 1D704,% lower case iota (maths italic)
12409   \glshex 1D6EA% upper case iota (maths italic)
12410 }
```

MathItalicKappa

```
12411 \newcommand*{\glxtrMathItalicKappa}{%
12412   \glshex 1D705% lower case kappa (maths italic)
12413   \string=\glshex 1D718,% lower case kappa variant (maths italic)
12414   \glshex 1D6EB% upper case kappa (maths italic)
12415 }
```

athItalicLambda

```
12416 \newcommand*{\glxtrMathItalicLambda}{%
12417   \glshex 1D706,% lower case lambda (maths italic)
12418   \glshex 1D6EC% upper case lambda (maths italic)
12419 }
```

xtrMathItalicMu

```
12420 \newcommand*{\glxtrMathItalicMu}{%
12421   \glshex 1D707,% lower case mu (maths italic)
12422   \glshex 1D6ED% upper case mu (maths italic)
12423 }
```

xtrMathItalicNu

```
12424 \newcommand*{\glxtrMathItalicNu}{%
12425   \glshex 1D708,% lower case nu (maths italic)
```

```

12426 \glshex 1D6EE% upper case nu (maths italic)
12427 }

xtrMathItalicXi
12428 \newcommand*{\glsxtrMathItalicXi}{%
12429 \glshex 1D709,% lower case xi (maths italic)
12430 \glshex 1D6EF% upper case xi (maths italic)
12431 }

thItalicOmicron
12432 \newcommand*{\glsxtrMathItalicOmicron}{%
12433 \glshex 1D70A,% lower case omicron (maths italic)
12434 \glshex 1D6F0% upper case omicron (maths italic)
12435 }

xtrMathItalicPi
12436 \newcommand*{\glsxtrMathItalicPi}{%
12437 \glshex 1D70B% lower case pi (maths italic)
12438 \string=\glshex 1D71B,% lower case pi variant (maths italic)
12439 \glshex 1D6F1% upper case pi (maths italic)
12440 }

trMathItalicRho
12441 \newcommand*{\glsxtrMathItalicRho}{%
12442 \glshex 1D70C% lower case rho (maths italic)
12443 \string=\glshex 1D71A,% lower case rho variant (maths italic)
12444 \glshex 1D6F2% upper case rho (maths italic)
12445 }

MathItalicSigma
12446 \newcommand*{\glsxtrMathItalicSigma}{%
12447 \glshex 1D70D% lower case final sigma (maths italic)
12448 \string=\glshex 1D70E,% lower case sigma (maths italic)
12449 \glshex 1D6F4% upper case sigma (maths italic)
12450 }

trMathItalicTau
12451 \newcommand*{\glsxtrMathItalicTau}{%
12452 \glshex 1D70F,% lower case tau (maths italic)
12453 \glshex 1D6F5% upper case tau (maths italic)
12454 }

thItalicUpsilon
12455 \newcommand*{\glsxtrMathItalicUpsilon}{%
12456 \glshex 1D710,% lower case upsilon (maths italic)
12457 \glshex 1D6F6% upper case upsilon (maths italic)
12458 }

```

trMathItalicPhi

```
12459 \newcommand*{\glxtrMathItalicPhi}{%
12460   \glshex 1D711% lower case phi (maths italic)
12461   \string=\glshex 1D719,% lower case phi variant (maths italic)
12462   \glshex 1D6F7% upper case phi (maths italic)
12463 }
```

trMathItalicChi

```
12464 \newcommand*{\glxtrMathItalicChi}{%
12465   \glshex 1D712,% lower case chi (maths italic)
12466   \glshex 1D6F8% upper case chi (maths italic)
12467 }
```

trMathItalicPsi

```
12468 \newcommand*{\glxtrMathItalicPsi}{%
12469   \glshex 1D713,% lower case psi (maths italic)
12470   \glshex 1D6F9% upper case psi (maths italic)
12471 }
```

MathItalicOmega

```
12472 \newcommand*{\glxtrMathItalicOmega}{%
12473   \glshex 1D714,% lower case omega (maths italic)
12474   \glshex 1D6FA% upper case omega (maths italic)
12475 }
```

thItalicPartial

```
12476 \newcommand*{\glxtrMathItalicPartial}{%
12477   \glshex 1D715% partial differential (maths italic)
12478 }
```

MathItalicNabla

```
12479 \newcommand*{\glxtrMathItalicNabla}{%
12480   \glshex 1D6FB% nabla (maths italic)
12481 }
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
12482 \newcommand*{\glxtrdigitrules}{%
12483   0\string=\glshex 2080\string=\glshex 2070
12484   \string<1\string=\glshex 2081\string=\glshex 00B9
12485   \string<2\string=\glshex 2082\string=\glshex 00B2
12486   \string<3\string=\glshex 2083\string=\glshex 00B3
12487   \string<4\string=\glshex 2084\string=\glshex 2074
12488   \string<5\string=\glshex 2085\string=\glshex 2075
12489   \string<6\string=\glshex 2086\string=\glshex 2076
12490   \string<7\string=\glshex 2087\string=\glshex 2077
12491   \string<8\string=\glshex 2088\string=\glshex 2078
12492   \string<9\string=\glshex 2089\string=\glshex 2079
12493 }
```

BasicDigitrules Digits from the Basic Latin set.

```
12494 \newcommand*{\glxtrBasicDigitrules}{%
12495 0\string<1\string<2\string<3\string<4%
12496 \string<5\string<6\string<7\string<8\string<9%
12497 }
```

criptDigitrules Subscript digits.

```
12498 \newcommand*{\glxtrSubScriptDigitrules}{%
12499 \glshex 2080% subscript 0
12500 \string<\glshex 2081% subscript 1
12501 \string<\glshex 2082% subscript 2
12502 \string<\glshex 2083% subscript 3
12503 \string<\glshex 2084% subscript 4
12504 \string<\glshex 2085% subscript 5
12505 \string<\glshex 2086% subscript 6
12506 \string<\glshex 2087% subscript 7
12507 \string<\glshex 2088% subscript 8
12508 \string<\glshex 2089% subscript 9
12509 }
```

criptDigitrules Superscript digits.

```
12510 \newcommand*{\glxtrSuperScriptDigitrules}{%
12511 \glshex 2070% superscript 0
12512 \string<\glshex 00B9% superscript 1
12513 \string<\glshex 00B2% superscript 2
12514 \string<\glshex 00B3% superscript 3
12515 \string<\glshex 2074% superscript 4
12516 \string<\glshex 2075% superscript 5
12517 \string<\glshex 2076% superscript 6
12518 \string<\glshex 2077% superscript 7
12519 \string<\glshex 2078% superscript 8
12520 \string<\glshex 2079% superscript 9
12521 }
```

trfractionrules Vulgar fractions.

```
12522 \newcommand*{\glxtrfractionrules}{%
12523 \glshex 215F% fraction numerator one (1/)
12524 \string<\glshex 2189% zero thirds (0/3 = 0)
12525 \string<\glshex 2152% one tenth (1/10 = 0.1)
12526 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
12527 \string<\glshex 215B% one eighth (1/8 = 0.125)
12528 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
12529 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
12530 \string<\glshex 2155% one fifth (1/5 = 0.2)
12531 \string<\glshex 00BC% one quarter (1/4 = 0.25)
12532 \string<\glshex 2153% one third (1/3 ~ 0.333)
12533 \string<\glshex 215C% three eighths (3/8 = 0.375)
12534 \string<\glshex 2156% two fifths (2/5 = 0.4)
12535 \string<\glshex 00BD% one half (1/2 = 0.5)
```

```

12536 \string<\glshex 2157% three fifths (3/5 = 0.6)
12537 \string<\glshex 215D% five eighths (5/8 = 0.625)
12538 \string<\glshex 2154% two thirds (2/3 ~ 0.667)
12539 \string<\glshex 00BE% three quarters (3/4 = 0.75)
12540 \string<\glshex 2158% four fifths (4/5 = 0.8)
12541 \string<\glshex 215A% five sixths (5/6 ~ 0.833)
12542 \string<\glshex 215E% seven eighths (7/8 = 0.875)
12543 }

```

sxtrdialecthook Check for scripts associated with the document dialects.

```

12544 \renewcommand{\@glxsxtrdialecthook}{%
12545   \ifundef\CurrentTrackedScript
12546   {%
12547     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
12548     {%
12549       \edef\CurrentTrackedScript{%
12550         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
12551       }%
12552     }%
12553   }%
12554   {}%
12555   \ifdef\CurrentTrackedScript
12556   {%
12557     \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix
12558     \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
12559     \let\CurrentTrackedTag\CurrentTrackedScript
12560     \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
12561     {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
12562     {}%
12563     \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
12564   }%
12565   {}%
12566 }

```

If `\glsxtr@loaddialect` has been defined, then `glossaries-extra-bib2gls` has been loaded after `glossaries-extra`. (For example, through `\glossariesextrasetup`.) Not recommended, but if this has been done try to find the associated language resources.

```

12567 \ifdef\glsxtr@loaddialect
12568 {%
12569   \@ifpackageloaded{tracklang}
12570   {%
12571     \AnyTrackedLanguages
12572     {%
12573       \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
12574     }%
12575   }%
12576 }
12577 {}
12578 }

```

12579 {}

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:

```
12580 \NeedsTeXFormat{LaTeX2e}
12581 \ProvidesPackage{glossaries-extra-stylemods}[2018/04/09 v1.29 (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`.

`glxtr@loadstyles`

```
12582 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
12583 \DeclareOption{all}{%
12584   \appto\@glxtr@loadstyles{%
12585     \RequirePackage{glossary-inline}%
12586     \RequirePackage{glossary-list}%
12587     \RequirePackage{glossary-tree}%
12588     \RequirePackage{glossary-mcols}%
12589     \RequirePackage{glossary-long}%
12590     \RequirePackage{glossary-longragged}%
12591     \RequirePackage{glossary-longbooktabs}%
12592     \RequirePackage{glossary-super}%
12593     \RequirePackage{glossary-superragged}%
12594     \RequirePackage{glossary-bookindex}%
12595   }
12596 }

12597 \DeclareOption*{%
12598   \IfFileExists{glossary-\CurrentOption.sty}
12599   {\eappto\@glxtr@loadstyles{%
12600     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
12601   }%
12602   {%
12603     \PackageError{glossaries-extra-styles}%

```

```

12604     {Unknown option ‘\CurrentOption’}{}%
12605   }%
12606 }

```

Process the package options:

```
12607 \ProcessOptions
```

Load the required packages:

```
12608 \@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`.

```
12609 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

`ewglossarystyle`

```

12610 \providecommand{\renewglossarystyle}[2]{%
12611   \ifcsundef{@glsstyle@#1}%
12612   {%
12613     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
12614   }%
12615   {%
12616     \csdef{@glsstyle@#1}{#2}%
12617   }%
12618 }

```

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.

```

12619 \ifdef{@glsstyle@listdotted}
12620 {%
12621   \renewglossarystyle{listdotted}{%
12622     \setglossarystyle{list}%
12623     \renewcommand*{\glossentry}[2]{%
12624       \item[]\makebox[\glslistdottedwidth][l]{%
12625         \glsentryitem{##1}%
12626         \glstarget{##1}{\glossentryname{##1}}%
12627         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
12628         \glossentrydesc{##1}\glspostdescription}%
12629     \renewcommand*{\subglossentry}[3]{%
12630       \item[]\makebox[\glslistdottedwidth][l]{%
12631         \glssubentryitem{##2}%
12632         \glstarget{##2}{\glossentryname{##2}}%
12633         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
12634         \glossentrydesc{##2}\glspostdescription}%
12635   }

```

```
12636 }
12637 {%
```

Assume the style isn't required if it hasn't already been defined.

```
12638 }
```

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.

```
12639 \ifdef{\@glsstyle@list}
12640 {%
```

listprelocation Space before number list for top-level entries.

```
12641 \newcommand{\glslistprelocation}{\glsxtrprelocation}
```

childprelocation Space before number list for child entries.

```
12642 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
12643 \newcommand{\glslistchildpostlocation}{.}
```

Redefine list to use these commands.

```
12644 \renewglossarystyle{list}{%
12645   \renewenvironment{theglossary}%
12646     {\begin{description}}{\end{description}}%
12647   \renewcommand*\glossaryheader{}%
12648   \renewcommand*\glsgroupheading}[1]{%
12649     \renewcommand*\glossentry}[2]{%
12650       \item[\glsentryitem{##1}]%
12651         \glstarget{##1}{\glossentryname{##1}}]
12652       \glossentrydesc{##1}\glspostdescription\glslistprelocation ##2}%
12653   \renewcommand*\subglossentry}[3]{%
12654     \glssubentryitem{##2}%
12655     \glstarget{##2}{\strut}\space
12656     \glossentrydesc{##2}\glspostdescription
12657     \glslistchildprelocation ##3\glslistchildpostlocation}%
12658   \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\indexspace\fi}%
12659 }
12660 }
12661 }
```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```
12662 \ifdef{\@glsstyle@altlist}
12663 {%
12664   \renewglossarystyle{altlist}{%
12665     \setglossarystyle{list}%
12666     \renewcommand*\glossentry}[2]{%
12667       \item[\glsentryitem{##1}]%
```

```

12668      \glstarget{##1}{\glossentryname{##1}}}%
12669      \mbox{}\par\nobreak\@afterheading
12670      \glossentrydesc{##1}\glspostdescription\glslistprelocation ##2}%
12671  \renewcommand{\subglossentry}[3]{%
12672    \par
12673    \glssubentryitem{##2}%
12674    \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
12675    \glslistchildprelocation ##3}%
12676  }
12677 }
12678 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

12679 \ifdef{\@glsstyle@listgroup}
12680 {%
12681   \renewglossarystyle{listgroup}{%
12682     \setglossarystyle{list}%
12683     \renewcommand*\glsgroupheading[1]{%
12684       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}}%
12685       \mbox{}\par\nobreak\@afterheading
12686     }%
12687   }
12688 }
12689 {}

```

Similarly for listhypergroup.

```

12690 \ifdef{\@glsstyle@listhypergroup}
12691 {%
12692   \renewglossarystyle{listhypergroup}{%
12693     \setglossarystyle{list}%
12694     \renewcommand*\glossaryheader{%
12695       \glslistnavigationitem{\glsnavigation}}%
12696     \renewcommand*\glsgroupheading[1]{%
12697       \item[\glslistgroupheaderfmt
12698         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
12699       \mbox{}\par\nobreak\@afterheading
12700     }%
12701   }
12702 }
12703 {}

```

Similarly for altlistgroup.

```

12704 \ifdef{\@glsstyle@altlistgroup}
12705 {%
12706   \renewglossarystyle{altlistgroup}{%
12707     \setglossarystyle{altlist}%
12708     \renewcommand*\glsgroupheading[1]{%
12709       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}}%
12710       \mbox{}\par\nobreak\@afterheading
12711     }%
12712   }

```

```

12713 }
12714 {}

    Similarly for altlisthypergroup.
12715 \ifdef{\@glsstyle@altlisthypergroup}
12716 {%
12717   \renewglossarystyle{altlisthypergroup}{%
12718     \setglossarystyle{altlist}%
12719     \renewcommand*\{glossaryheader\}{%
12720       \glslistnavigationitem{\glsnavigation}}%
12721     \renewcommand*\{glsgroupheading}[1]{%
12722       \item[\glslistgroupheaderfmt
12723         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
12724       \mbox{}\par\nobreak\@afterheading
12725     }%
12726   }
12727 }
12728 {}

```

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`.

```

12729 \ifcsdef{@glsstyle@long}
12730 {%
12731   \renewglossarystyle{long}{%
12732     \renewenvironment{theglossary}%
12733       {\begin{longtable}{lp{\glsdescwidth}}}%
12734       {\end{longtable}}%
12735     \renewcommand*\{glossaryheader\}{}%
12736     \renewcommand*\{glsgroupheading}[1]{}%
12737     \renewcommand{\glossentry}[2]{%
12738       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12739       \glossentrydesc{##1}\glspostdescription
12740       \glstrprelocation ##2\tabularnewline
12741     }%
12742     \renewcommand{\subglossentry}[3]{%
12743       &
12744       \glssubentryitem{##2}%
12745       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
12746       \glstrprelocation ##3\tabularnewline
12747     }%
12748     \ifglsnogroupskip
12749       \renewcommand*\{glsgroupskip\}{}%
12750     \else
12751       \renewcommand*\{glsgroupskip\}{ & \tabularnewline}%
12752     \fi
12753   }

```

```
12754 }
12755 {}
```

Three column style:

```
12756 \ifcsdef{@glsstyle@long3col}
12757 {%
12758   \renewglossarystyle{long3col}{%
12759     \renewenvironment{theglossary}%
12760       {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
12761       {\end{longtable}}}%
12762     \renewcommand*{\glossaryheader}{}%
12763     \renewcommand*{\glsgroupheading}[1]{}%
12764     \renewcommand{\glossentry}[2]{%
12765       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12766       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
12767     }%
12768     \renewcommand{\subglossentry}[3]{%
12769       &
12770       \glssubentryitem{##2}%
12771       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12772       ##3\tabularnewline
12773     }%
12774   }
```

Conditional needs to be outside of \glsgroupskip otherwise it can cause “Incomplete \iftrue” errors.

```
12774   \ifglsgroupskip
12775     \renewcommand*{\glsgroupskip}{}%
12776   \else
12777     \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
12778   \fi
12779 }
12780 }
12781 {}
```

Four column style:

```
12782 \ifcsdef{@glsstyle@long4col}
12783 {%
12784   \renewglossarystyle{long4col}{%
12785     \renewenvironment{theglossary}%
12786       {\begin{longtable}{llll}}}%
12787       {\end{longtable}}}%
12788     \renewcommand*{\glossaryheader}{}%
12789     \renewcommand*{\glsgroupheading}[1]{}%
12790     \renewcommand{\glossentry}[2]{%
12791       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12792       \glossentrydesc{##1}\glspostdescription &
12793       \glossentrysymbol{##1} &
12794       ##2\tabularnewline
12795     }%
12796     \renewcommand{\subglossentry}[3]{%
12797       &
```

```

12798      \glssubentryitem{##2}%
12799      \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12800      \glossentrysymbol{##2} & ##3\tabularnewline
12801  }%

12802  \ifglsnogroupskip
12803      \renewcommand*{\glsgroupskip}{}%
12804  \else
12805      \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
12806  \fi
12807  }
12808  }
12809  {}

```

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`.

```

12810 \ifcsdef{@glstyle@longragged}
12811 {%
12812   \renewglossarystyle{longragged}{%
12813     \renewenvironment{theglossary}%
12814       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}}}%
12815       {\end{longtable}}%
12816     \renewcommand*{\glossaryheader}{}%
12817     \renewcommand*{\glsgroupheading}[1]{}%
12818     \renewcommand{\glossentry}[2]{%
12819       \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12820       \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
12821       \tabularnewline
12822     }%
12823     \renewcommand{\subglossentry}[3]{%
12824       &
12825       \glssubentryitem{##2}%
12826       \glstarget{##2}{\strut}\glossentrydesc{##2}%
12827       \glspostdescription\glxtrprelocation ##3%
12828       \tabularnewline
12829     }%
12830   \ifglsnogroupskip
12831     \renewcommand*{\glsgroupskip}{}%
12832   \else
12833     \renewcommand*{\glsgroupskip}{& \tabularnewline}%
12834   \fi
12835 }
12836 }

```

12837 {}

Three and four column styles don't use `\glxtrprelocation` since the number list is in its own column.

```
12838 \ifcsdef{@glsstyle@longragged3col}
12839 {%
12840   \renewglossarystyle{longragged3col}{%
12841     \renewenvironment{theglossary}%
12842       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
12843         >{\raggedright}p{\glspagelistwidth}}}%
12844       {\end{longtable}}}%
12845     \renewcommand*{\glossaryheader}{}%
12846     \renewcommand*{\glsgroupheading}[1]{}%
12847     \renewcommand{\glossentry}[2]{%
12848       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12849       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
12850     }%
12851     \renewcommand{\subglossentry}[3]{%
12852       &
12853       \glssubentryitem{##2}%
12854       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12855       ##3\tabularnewline
12856     }%
12857     \ifglsgroupskip
12858       \renewcommand*{\glsgroupskip}{}%
12859     \else
12860       \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
12861     \fi
12862   }
12863 }
12864 {}
```

Four column style:

```
12865 \ifcsdef{@glsstyle@altlongragged4col}
12866 {%
12867   \renewglossarystyle{altlongragged4col}{%
12868     \renewenvironment{theglossary}%
12869       {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
12870         >{\raggedright}p{\glspagelistwidth}}}%
12871       {\end{longtable}}}%
12872     \renewcommand*{\glossaryheader}{}%
12873     \renewcommand*{\glsgroupheading}[1]{}%
12874     \renewcommand{\glossentry}[2]{%
12875       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12876       \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
12877       ##2\tabularnewline
12878     }%
12879     \renewcommand{\subglossentry}[3]{%
12880       &
```

```

12881     \glssubentryitem{##2}%
12882     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12883     \glossentrysymbol{##2} & ##3\tabularnewline
12884 }%

12885 \ifglsnogroupskip
12886     \renewcommand*{\glsgroupskip}{}%
12887 \else
12888     \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
12889 \fi
12890 }
12891 }
12892 {}

```

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`.

```

12893 \ifcsdef{@glstyle@super}
12894 {%
12895     \renewglossarystyle{super}{%
12896         \renewenvironment{theglossary}%
12897             {\tablehead{}\tabletail}%
12898             \begin{supertabular}{lp{\glsdescwidth}}%
12899             {\end{supertabular}}%
12900     \renewcommand*{\glossaryheader}{}%
12901     \renewcommand*{\glsgroupheading}[1]{}%
12902     \renewcommand{\glossentry}[2]{%
12903         \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12904         \glossentrydesc{##1}\glspostdescription
12905         \glxtrprelocation ##2\tabularnewline
12906     }%
12907     \renewcommand{\subglossentry}[3]{%
12908         &
12909         \glssubentryitem{##2}%
12910         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
12911         \glxtrprelocation ##3\tabularnewline
12912     }%
12913     \ifglsnogroupskip
12914         \renewcommand*{\glsgroupskip}{}%
12915     \else
12916         \renewcommand*{\glsgroupskip}{& \tabularnewline}%
12917     \fi
12918 }
12919 }
12920 {}

```

Three column style:

```

12921 \ifcsdef{@glstyle@super3col}

```

```

12922 {%
12923   \renewglossarystyle{super3col}{%
12924     \renewenvironment{theglossary}%
12925       {\tablehead{}\tabletail{}}%
12926       \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
12927     {\end{supertabular}}%
12928     \renewcommand*{\glossaryheader}{}%
12929     \renewcommand*{\glsgroupheading}[1]{}%
12930     \renewcommand{\glossentry}[2]{%
12931       \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12932       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
12933     }%
12934     \renewcommand{\subglossentry}[3]{%
12935       &
12936       \glssubentryitem{##2}%
12937       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12938       ##3\tabularnewline
12939     }%

12940     \ifglsgnogroupskip
12941       \renewcommand*{\glsgroupskip}{}%
12942     \else
12943       \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
12944     \fi
12945   }
12946 }
12947 {}

```

Four column styles:

```

12948 \ifcsdef{@glssstyle@super4col}
12949 {%
12950   \renewglossarystyle{super4col}{%
12951     \renewenvironment{theglossary}%
12952       {\tablehead{}\tabletail{}}%
12953       \begin{supertabular}{llll}}{%
12954     \end{supertabular}}%
12955     \renewcommand*{\glossaryheader}{}%
12956     \renewcommand*{\glsgroupheading}[1]{}%
12957     \renewcommand{\glossentry}[2]{%
12958       \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12959       \glossentrydesc{##1}\glspostdescription &
12960       \glossentrysymbol{##1} & ##2\tabularnewline
12961     }%
12962     \renewcommand{\subglossentry}[3]{%
12963       &
12964       \glssubentryitem{##2}%
12965       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
12966       \glossentrysymbol{##2} & ##3\tabularnewline
12967     }%

```

```

12968 \ifglsgroupskip
12969 \renewcommand*{\glsgroupskip}{}%
12970 \else
12971 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
12972 \fi
12973 }
12974 }
12975 {}

```

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`.

```

12976 \ifcsdef{@glstyle@superragged}
12977 {%
12978 \renewglossarystyle{superragged}{%
12979 \renewenvironment{theglossary}%
12980 {\tablehead{}\tabletail{}}%
12981 \begin{supertabular}{1>{\raggedright}p{\glsgdescwidth}}%
12982 {\end{supertabular}}%
12983 \renewcommand*{\glossaryheader}{}%
12984 \renewcommand*{\glsgroupheading}[1]{}%
12985 \renewcommand{\glossentry}[2]{%
12986 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
12987 \glossentrydesc{##1}\glspostdescription\glstrprelocation ##2%
12988 \tabularnewline
12989 }%
12990 \renewcommand{\subglossentry}[3]{%
12991 &
12992 \glssubentryitem{##2}%
12993 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
12994 \glstrprelocation ##3%
12995 \tabularnewline
12996 }%
12997 \ifglsgroupskip
12998 \renewcommand*{\glsgroupskip}{}%
12999 \else
13000 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
13001 \fi
13002 }
13003 }
13004 {}

```

Three column style:

```

13005 \ifcsdef{@glstyle@superragged3col}
13006 {%
13007 \renewglossarystyle{superragged3col}{%

```

```

13008 \renewenvironment{theglossary}%
13009 {\tablehead{}\tabletail{}}%
13010 \begin{supertabular}{1>\raggedrightp{\glsgdescwidth}%
13011 >\raggedrightp{\glspagelistwidth}}}%
13012 {\end{supertabular}}%
13013 \renewcommand*\glossaryheader{}%
13014 \renewcommand*\glsgroupheading}[1]{}%
13015 \renewcommand{\glossentry}[2]{%
13016 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13017 \glossentrydesc{##1}\glspostdescription &
13018 ##2\tabularnewline
13019 }%
13020 \renewcommand{\subglossentry}[3]{%
13021 &
13022 \glssubentryitem{##2}%
13023 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13024 ##3\tabularnewline
13025 }%

13026 \ifglsgroupskip
13027 \renewcommand*\glsgroupskip{}%
13028 \else
13029 \renewcommand*\glsgroupskip{ & \tabularnewline}%
13030 \fi
13031 }
13032 }
13033 {}

```

Four columns:

```

13034 \ifcsdef{@glstyle@altsuperragged4col}
13035 {%
13036 \renewglossarystyle{altsuperragged4col}{%
13037 \renewenvironment{theglossary}%
13038 {\tablehead{}\tabletail{}}%
13039 \begin{supertabular}{1>\raggedrightp{\glsgdescwidth}1%
13040 >\raggedrightp{\glspagelistwidth}}}%
13041 {\end{supertabular}}%
13042 \renewcommand*\glossaryheader{}%
13043 \renewcommand{\glossentry}[2]{%
13044 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
13045 \glossentrydesc{##1}\glspostdescription &
13046 \glossentrysymbol{##1} & ##2\tabularnewline
13047 }%
13048 \renewcommand{\subglossentry}[3]{%
13049 &
13050 \glssubentryitem{##2}%
13051 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
13052 \glossentrysymbol{##2} & ##3\tabularnewline
13053 }%

```

```

13054 \ifglsnogroupskip
13055 \renewcommand*{\glsgroupskip}{}%
13056 \else
13057 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
13058 \fi
13059 }
13060 }
13061 {}

```

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`.

```

13062 \ifdef{\@glsstyle@inline}
13063 {%
13064 \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
13065 \renewcommand*{\glsinlinedescformat}[3]{%
13066 \space#1\glxtrpostdescription}
13067 \renewcommand*{\glsinlinesubdescformat}[3]{%
13068 #1\glxtrpostdescription}

```

Just use `\glxtrpostdescription` instead of `\glspostdescription`.

```

13069 }
13070 {}

```

2.8 Tree Styles

The index style is redefined so that the space before the number list isn't hard coded.

```

13071 \ifdef{\@glsstyle@index}
13072 {

```

`treeprelocation` The space before the number list for top-level entries. This is shared by the other tree styles.

```

13073 \newcommand*{\glstreeprelocation}{\glxtrprelocation}

```

`childprelocation` The space before the number list for child entries. This is shared by the other tree styles.

```

13074 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}

```

```

13075 \renewglossarystyle{index}{%
13076 \renewenvironment{theglossary}%
13077 {\setlength{\parindent}{0pt}%
13078 \setlength{\parskip}{0pt plus 0.3pt}%
13079 \let\item\glstreeitem
13080 \let\subitem\glstreesubitem}

```

```

13081      \let\subsubitem\glstreesubsubitem
13082    }%
13083  {\par}%
13084  \renewcommand*{\glossaryheader}{}%
13085  \renewcommand*{\glsgroupheading}[1]{}%
13086  \renewcommand*{\glossentry}[2]{%
13087    \item\glstentryitem{##1}%
13088    \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13089    \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
13090    \glstreepredesc \glossentrydesc{##1}\glspostdescription
13091    \glstreeprelocation ##2%
13092  }%
13093  \renewcommand{\subglossentry}[3]{%
13094    \ifcase##1\relax
13095      \item
13096    \or
13097      \subitem
13098      \glssubentryitem{##2}%
13099    \else
13100      \subsubitem
13101    \fi
13102    \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13103    \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
13104    \glstreechildpredesc\glossentrydesc{##2}\glspostdescription
13105    \glstreechildprelocation ##3%
13106  }%
13107  \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13108 }
13109 }
13110 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

13111 \ifdef{\@glsstyle@indexgroup}
13112 {%
13113   \renewglossarystyle{indexgroup}{%
13114     \setglossarystyle{index}%
13115     \renewcommand*{\glsgroupheading}[1]{%
13116       \item\glstreegroupheaderfmt{\glsggetgrouptitle{##1}}%
13117       \nopagebreak\indexspace
13118       \nobreak\@afterheading
13119     }%
13120   }
13121 }
13122 {}

```

Similarly for indexhypergroup.

```

13123 \ifdef{\@glsstyle@indexhypergroup}
13124 {%
13125   \renewglossarystyle{indexhypergroup}{%
13126     \setglossarystyle{index}%

```

```

13127 \renewcommand*{\glossaryheader}{%
13128 \item\glstreenavigationfmt{\glsnavigation}%
13129 \nobreak\@afterheading\indexspace}%
13130 \renewcommand*{\glsgroupheading}[1]{%
13131 \item\glstreegroupheaderfmt
13132 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13133 \nopagebreak\indexspace
13134 \nobreak\@afterheading}%
13135 }%
13136 }
13137 {}

```

Adjust tree style to remove hard coded space before number list.

```

13138 \ifdef{\@glsstyle@tree}
13139 {%
13140 \renewglossarystyle{tree}{%
13141 \renewenvironment{theglossary}%
13142 {\setlength{\parindent}{0pt}%
13143 \setlength{\parskip}{0pt plus 0.3pt}}%
13144 {}%
13145 \renewcommand*{\glossaryheader}{}%
13146 \renewcommand*{\glsgroupheading}[1]{}%
13147 \renewcommand{\glossentry}[2]{%
13148 \hangindent0pt\relax
13149 \parindent0pt\relax
13150 \glstryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13151 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}%
13152 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13153 \glstreeprelocation##2\par
13154 }%
13155 \renewcommand{\subglossentry}[3]{%
13156 \hangindent##1\glstreeindent\relax
13157 \parindent##1\glstreeindent\relax
13158 \ifnum##1=1\relax
13159 \glssubentryitem{##2}%
13160 \fi
13161 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
13162 \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}%
13163 \glstreechildpredesc\glossentrydesc{##2}\glspostdescription
13164 \glstreechildprelocation ##3\par
13165 }%
13166 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13167 }%
13168 }
13169 {}

```

The treegroup style is redefined to discourage a page break after the heading.

```

13170 \ifdef{\@glsstyle@treegroup}
13171 {%
13172 \renewglossarystyle{treegroup}{%

```

```

13173 \setglossarystyle{tree}%
13174 \renewcommand{\glsgroupheading}[1]{\par
13175 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
13176 \nopagebreak\indexspace\nobreak\@afterheading}%
13177 }
13178 }
13179 {}

```

Similarly for treehypergroup

```

13180 \ifdef{\@glsstyle@treehypergroup}
13181 {%
13182 \renewglossarystyle{treehypergroup}{%
13183 \setglossarystyle{tree}%
13184 \renewcommand*\{\glossaryheader}{%
13185 \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13186 \nobreak\@afterheading\indexspace}%
13187 \renewcommand*\{\glsgroupheading}[1]{%
13188 \par\noindent
13189 \glstreegroupheaderfmt
13190 {\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
13191 \nopagebreak\indexspace\nobreak\@afterheading}%
13192 }
13193 }
13194 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

13195 \ifdef{\@glsstyle@treenoname}
13196 {%
13197 \renewglossarystyle{treenoname}{%
13198 \renewenvironment{theglossary}%
13199 {\setlength{\parindent}{0pt}%
13200 \setlength{\parskip}{0pt plus 0.3pt}}%
13201 {}%
13202 \renewcommand*\{\glossaryheader}{}%
13203 \renewcommand*\{\glsgroupheading}[1]{}%
13204 \renewcommand{\glossentry}[2]{%
13205 \hangindent0pt\relax
13206 \parindent0pt\relax
13207 \glssentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13208 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
13209 \glstreepredesc\glossentrydesc{##1}\glspostdescription
13210 \glstreeprelocation##2\par
13211 }%
13212 \renewcommand{\subglossentry}[3]{%
13213 \hangindent##1\glstreeindent\relax
13214 \parindent##1\glstreeindent\relax
13215 \ifnum##1=1\relax
13216 \glssubentryitem{##2}%
13217 \fi
13218 \glstarget{##2}{\strut}%

```

```

13219      \glossentrydesc{##2}\glspostdescription\glstreechildprelocation##3\par
13220    }%
13221    \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13222  }
13223 }
13224 {}

```

The `treenonamegroup` style is redefined to discourage a page break after the heading.

```

13225 \ifdef{\@glsstyle@treenonamegroup}
13226 {%
13227   \renewglossarystyle{treenonamegroup}{%
13228     \setglossarystyle{treenoname}%
13229     \renewcommand{\glsgroupheading}[1]{\par
13230       \noindent\glstreegroupheaderfmt
13231       {\glsgrouptitle{##1}}}%
13232     \nopagebreak\indexspace\nobreak\@afterheading
13233   }%
13234 }
13235 }
13236 {}

```

Similarly for `treenonamehypergroup`

```

13237 \ifdef{\@glsstyle@treenonamehypergroup}
13238 {%
13239   \renewglossarystyle{treenonamehypergroup}{%
13240     \setglossarystyle{treenoname}%
13241     \renewcommand*{\glossaryheader}{%
13242       \par\noindent\glstreenavigationfmt{\glsnavigation}\par
13243       \nobreak\@afterheading\indexspace}%
13244     \renewcommand*{\glsgroupheading}[1]{%
13245       \par\noindent
13246       \glstreegroupheaderfmt
13247       {\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}%
13248     \nopagebreak\indexspace\nobreak\@afterheading}%
13249   }
13250 }
13251 {}

```

The `almtree` style is redefined to make it easier to make minor adjustments.

```

13252 \ifdef{\@glsstyle@almtree}
13253 {%

```

Only redefine this style if it's already been defined.

symbolDescLocation

```
\glxtralttreeSymbolDescLocation{\langle label \rangle}{\langle location list \rangle}
```

Layout the symbol, description and location for top-level entries.

```

13254   \newcommand{\glxtralttreeSymbolDescLocation}[2]{%

```

```

13255   {%
13256     \let\par\glxstrAltTreePar
13257     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
13258     \glossentrydesc{#1}\glspostdescription\glstreeprelocation #2\par
13259   }%
13260 }

```

trAltTreeIndent Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

13261 \newlength\glxstrAltTreeIndent

```

lsxtrAltTreePar Multi-paragraph descriptions need to keep the hanging indent.

```

13262 \newcommand{\glxstrAltTreePar}{%
13263   \@@par
13264   \glxstrAltTreeSetHangIndent
13265   \setlength{\parindent}{\dimexpr\hangindent+\glxstrAltTreeIndent}%
13266 }

```

symbolDescLocation `\glxstralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

13267 \newcommand{\glxstralttreeSubSymbolDescLocation}[3]{%
13268   \glxstralttreeSymbolDescLocation{#2}{#3}%
13269 }

```

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.

```

13270 \newlength\glxstrtreetopindent

```

lxstralttreeInit User-level initialisation for the alttree style.

```

13271 \newcommand*{\glxstralttreeInit}{%
13272   \settowidth{\glxstrtreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
13273   \glxstrAltTreeIndent=\parindent
13274 }

```

\glsglsetwidest The original `\glsglsetwidest` only uses `\def`. This uses `\gdef`.

```

13275 \newcommand*{\glsglsetwidest}[2][0]{%
13276   \csgdef{@glswidestname\romannumeral#1}{#2}%
13277 }

```

\eglssetwidest The original `\glsglsetwidest` only uses `\def`. This uses `\protected@csedef`.

```

13278 \newcommand*{\eglssetwidest}[2][0]{%
13279   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
13280 }

```

`\xglsetwidest` Like the above but uses `\protected@csxdef`.

```
13281 \newcommand*\xglsetwidest}[2][0]{%
13282   \protected@csxdef{#1}{#2}%
13283 }
```

`glsupdatewidest` Only sets if new value is wider than old value.

```
13284 \newcommand*\glsupdatewidest}[2][0]{%
13285   \ifcsundef{#1}%
13286   {\csdef{#1}{#2}}%
13287   {%
13288     \settowidth{#1}{\csuse{#1}}%
13289     \settowidth{#2}%
13290     \ifdim#2>#1%
13291       \csdef{#1}{#2}%
13292     \fi
13293   }%
13294 }
```

`glsupdatewidest` As above but global definition.

```
13295 \newcommand*\gglsupdatewidest}[2][0]{%
13296   \ifcsundef{#1}%
13297   {\csgdef{#1}{#2}}%
13298   {%
13299     \settowidth{#1}{\csuse{#1}}%
13300     \settowidth{#2}%
13301     \ifdim#2>#1%
13302       \csgdef{#1}{#2}%
13303     \fi
13304   }%
13305 }
```

`glsupdatewidest` As `\glsupdatewidest` but expands value.

```
13306 \newcommand*\eglsupdatewidest}[2][0]{%
13307   \ifcsundef{#1}%
13308   {\protected@csedef{#1}{#2}}%
13309   {%
13310     \settowidth{#1}{\csuse{#1}}%
13311     \settowidth{#2}%
13312     \ifdim#2>#1%
13313       \protected@csedef{#1}{#2}%
13314     \fi
13315   }%
13316 }
```

`glsupdatewidest` As above but global.

```
13317 \newcommand*\xglsupdatewidest}[2][0]{%
13318   \ifcsundef{#1}%
13319   {\protected@csxdef{#1}{#2}}%
13320   {%
```

```

13321      \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
13322      \settowidth{\dimen@ii}{#2}%
13323      \ifdim\dimen@ii>\dimen@
13324      \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
13325      \fi
13326  }%
13327  }

```

glswidestname Provide a user-level macro to obtain the widest top-level name.

```

13328  \newcommand*{\glswidestname}{\@glswidestname}

```

glswidestsubname Provide a user-level macro to obtain the widest sub-entry name.

```

13329  \newcommand*{\glswidestsubname}[1]{%
13330    \ifcsundef{@glswidestname\romannumeral#1}%
13331    {\@glswidestname}%
13332    {\csuse{@glswidestname\romannumeral#1}}%
13333  }

```

glswidestTopLevelName CamelCase is easier for long command names. Provide a CamelCase synonym of `\glswidestTopLevelName`.

```

13334  \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.

```

13335  \newrobustcmd*{\glswidestUsedTopLevelName}[1][\@glo@types]{%
13336    \dimen@=0pt\relax
13337    \glstmpplen=0pt\relax
13338    \foralllglossaries[#1]{\@gls@type}%
13339    {%
13340      \forglsentries[\@gls@type]{\@glo@label}%
13341      {%
13342        \ifglswidest{\@glo@label}%
13343        {%
13344          \ifglshasparent{\@glo@label}%
13345          {}%
13346          {%
13347            \settowidth{\dimen@}{%
13348              {\glstreenamfmt{\glstentryname{\@glo@label}}}%
13349              \ifdim\dimen@>\glstmpplen
13350              \glstmpplen=\dimen@
13351              \eglssetwidest{\glstentryname{\@glo@label}}%
13352            \fi
13353          }%
13354        }%
13355      }%
13356    }%
13357  }%
13358  }

```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

13359 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
13360 \dimen@=0pt\relax
13361 \gls@tmplen=0pt\relax
13362 \foralllglossaries[#1]{\@gls@type}%
13363 {%
13364 \forallglsentries[\@gls@type]{\@glo@label}%
13365 {%
13366 \ifglsused{\@glo@label}%
13367 {%
13368 \settowidth{\dimen@}%
13369 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13370 \ifdim\dimen@>\gls@tmplen
13371 \gls@tmplen=\dimen@
13372 \eglssetwidest{\glsentryname{\@glo@label}}%
13373 \fi
13374 }%
13375 }%
13376 }%
13377 }%
13378 }

```

ndWidestAnyName Like the above but doesn't check is the entry has been used.

```

13379 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
13380 \dimen@=0pt\relax
13381 \gls@tmplen=0pt\relax
13382 \foralllglossaries[#1]{\@gls@type}%
13383 {%
13384 \forallglsentries[\@gls@type]{\@glo@label}%
13385 {%
13386 \settowidth{\dimen@}%
13387 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13388 \ifdim\dimen@>\gls@tmplen
13389 \gls@tmplen=\dimen@
13390 \eglssetwidest{\glsentryname{\@glo@label}}%
13391 \fi
13392 }%
13393 }%
13394 }

```

estUsedLevelTwo This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```

13395 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
13396 \dimen@=0pt\relax
13397 \dimen@i=0pt\relax
13398 \dimen@ii=0pt\relax
13399 \foralllglossaries[#1]{\@gls@type}%
13400 {%

```

```

13401 \forglsentries[\@gls@type]{\@glo@label}%
13402 {%
13403   \ifglsused{\@glo@label}%
13404   {%
13405     \ifglsashasparent{\@glo@label}%
13406     {%
13407       \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
13408       \ifglsashasparent{\@glo@parent}%
13409       {%
13410         \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
13411         \ifglsashasparent{\@glo@parent}%
13412         {}%
13413         {%
13414           \settowidth{\gls@tmplen}%
13415             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
13416           \ifdim\gls@tmplen>\dimen@ii
13417             \dimen@ii=\gls@tmplen
13418             \eglssetwidest[2]{\glsentryname{\@glo@label}}%
13419           \fi
13420         }%
13421       }%
13422     {%
13423       \settowidth{\gls@tmplen}%
13424         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
13425       \ifdim\gls@tmplen>\dimen@i
13426         \dimen@i=\gls@tmplen
13427         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
13428       \fi
13429     }%
13430   }%
13431   {%
13432     \settowidth{\gls@tmplen}%
13433       {\glstreenamefmt{\glsentryname{\@glo@label}}}%
13434     \ifdim\gls@tmplen>\dimen@
13435       \dimen@=\gls@tmplen
13436       \eglssetwidest{\glsentryname{\@glo@label}}%
13437     \fi
13438   }%
13439 }%
13440 {}%
13441 }%
13442 }%
13443 }

```

`dWidestLevelTwo` This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

13444 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
13445   \dimen@=0pt\relax
13446   \dimen@i=0pt\relax
13447   \dimen@ii=0pt\relax

```

```

13448 \forall glossaries[#1]{\@gls@type}%
13449 {%
13450 \forall glsentries[\@gls@type]{\@glo@label}%
13451 {%
13452 \ifglshasparent{\@glo@label}%
13453 {%
13454 \edef\@glo@parent{\csuse{glo@\glstetoklabel{\@glo@label}@parent}}%
13455 \ifglshasparent{\@glo@parent}%
13456 {%
13457 \edef\@glo@parent{\csuse{glo@\glstetoklabel{\@glo@parent}@parent}}%
13458 \ifglshasparent{\@glo@parent}%
13459 {}%
13460 {%
13461 \settowidth{\gls@tmplen}%
13462 {\glstreenamfmt{\glstentryname{\@glo@label}}}%
13463 \ifdim\gls@tmplen>\dimen@ii
13464 \dimen@ii=\gls@tmplen
13465 \eglssetwidest[2]{\glstentryname{\@glo@label}}%
13466 \fi
13467 }%
13468 }%
13469 {%
13470 \settowidth{\gls@tmplen}%
13471 {\glstreenamfmt{\glstentryname{\@glo@label}}}%
13472 \ifdim\gls@tmplen>\dimen@i
13473 \dimen@i=\gls@tmplen
13474 \eglssetwidest[1]{\glstentryname{\@glo@label}}%
13475 \fi
13476 }%
13477 }%
13478 {%
13479 \settowidth{\gls@tmplen}%
13480 {\glstreenamfmt{\glstentryname{\@glo@label}}}%
13481 \ifdim\gls@tmplen>\dimen@
13482 \dimen@=\gls@tmplen
13483 \eglssetwidest{\glstentryname{\@glo@label}}%
13484 \fi
13485 }%
13486 }%
13487 }%
13488 }

```

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.

```

13489 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
13490 \dimen@=0pt\relax
13491 \gls@tmplen=0pt\relax
13492 #2=0pt\relax
13493 \forall glossaries[#1]{\@gls@type}%

```

```

13494   {%
13495     \forglsentries[\@gls@type]{\@glo@label}%
13496     {%
13497       \ifglsused{\@glo@label}%
13498       {%
13499         \settowidth{\dimen@}%
13500         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13501         \ifdim\dimen@>\gls@tmplen
13502           \gls@tmplen=\dimen@
13503           \eglssetwidest{\glsentryname{\@glo@label}}%
13504         \fi
13505         \settowidth{\dimen@}%
13506         {\glsentrysymbol{\@glo@label}}%
13507         \ifdim\dimen@>#2\relax
13508           #2=\dimen@
13509         \fi
13510       }%
13511     }%
13512   }%
13513 }%
13514 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

13515   \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
13516     \dimen@=0pt\relax
13517     \gls@tmplen=0pt\relax
13518     #2=0pt\relax
13519     \forallglossaries[#1]{\@gls@type}%
13520     {%
13521       \forglsentries[\@gls@type]{\@glo@label}%
13522       {%
13523         \settowidth{\dimen@}%
13524         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
13525         \ifdim\dimen@>\gls@tmplen
13526           \gls@tmplen=\dimen@
13527           \eglssetwidest{\glsentryname{\@glo@label}}%
13528         \fi
13529         \settowidth{\dimen@}%
13530         {\glsentrysymbol{\@glo@label}}%
13531         \ifdim\dimen@>#2\relax
13532           #2=\dimen@
13533         \fi
13534       }%
13535     }%
13536   }

```

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.

```

13537 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
13538   \dimen@=0pt\relax
13539   \gls@tmplen=0pt\relax
13540   #2=0pt\relax
13541   #3=0pt\relax
13542   \foralllglossaries[#1]{\@gls@type}%
13543   {%
13544     \forglsentries[\@gls@type]{\@glo@label}%
13545     {%
13546       \ifglsused{\@glo@label}%
13547       {%
13548         \settowidth{\dimen@}%
13549         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
13550         \ifdim\dimen@>\gls@tmplen
13551           \gls@tmplen=\dimen@
13552           \eglssetwidest{\glsentryname{\@glo@label}}%
13553         \fi
13554         \settowidth{\dimen@}%
13555         {\glsentrysymbol{\@glo@label}}%
13556         \ifdim\dimen@>#2\relax
13557           #2=\dimen@
13558         \fi
13559         \settowidth{\dimen@}%
13560         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
13561         \ifdim\dimen@>#3\relax
13562           #3=\dimen@
13563         \fi
13564       }%
13565     }%
13566   }%
13567 }%
13568 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

13569 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
13570   \dimen@=0pt\relax
13571   \gls@tmplen=0pt\relax
13572   #2=0pt\relax
13573   #3=0pt\relax
13574   \foralllglossaries[#1]{\@gls@type}%
13575   {%
13576     \forglsentries[\@gls@type]{\@glo@label}%
13577     {%
13578       \settowidth{\dimen@}%
13579       {\glstreenamefmt{\glsentryname{\@glo@label}}}%
13580       \ifdim\dimen@>\gls@tmplen
13581         \gls@tmplen=\dimen@
13582         \eglssetwidest{\glsentryname{\@glo@label}}%

```

```

13583      \fi
13584      \settowidth{\dimen@}%
13585        {\glstrysymbol{\@glo@label}}}%
13586      \ifdim\dimen@>#2\relax
13587        #2=\dimen@
13588      \fi
13589      \settowidth{\dimen@}%
13590        {\GlsXtrFormatLocationList{\glstrynumberlist{\@glo@label}}}%
13591      \ifdim\dimen@>#3\relax
13592        #3=\dimen@
13593      \fi
13594    }%
13595  }%
13596 }

```

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.

```

13597 \newrobustcmd*{\glstryFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
13598   \dimen@=0pt\relax
13599   \glstry@tmplen=0pt\relax
13600   #2=0pt\relax
13601   \forallglossaries[#1]{\@gls@type}%
13602   {%
13603     \forallglsentries[\@gls@type]{\@glo@label}%
13604     {%
13605       \ifglstryused{\@glo@label}%
13606       {%
13607         \settowidth{\dimen@}%
13608           {\glstrynamefmt{\glstryname{\@glo@label}}}%
13609         \ifdim\dimen@>\glstry@tmplen
13610           \glstry@tmplen=\dimen@
13611           \eglstrysetwidest{\glstryname{\@glo@label}}%
13612         \fi
13613         \settowidth{\dimen@}%
13614           {\GlsXtrFormatLocationList{\glstrynumberlist{\@glo@label}}}%
13615         \ifdim\dimen@>#2\relax
13616           #2=\dimen@
13617         \fi
13618       }%
13619     }%
13620   }%
13621 }%
13622 }

```

AnyNameLocation Like the `\glstryFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

13623 \newrobustcmd*{\glstryFindWidestAnyNameLocation}[2][\@glo@types]{%
13624   \dimen@=0pt\relax
13625   \glstry@tmplen=0pt\relax

```

```

13626      #2=Opt\relax
13627      \forallglossaries[#1]{\@gls@type}%
13628      {%
13629        \forglsentries[\@gls@type]{\@glo@label}%
13630        {%
13631          \settowidth{\dimen@}%
13632            {\glstreenamefmt{\glsentryname{\@glo@label}}}%
13633          \ifdim\dimen@>\gls@tmplen
13634            \gls@tmplen=\dimen@
13635            \eglssetwidest{\glsentryname{\@glo@label}}%
13636          \fi
13637          \settowidth{\dimen@}%
13638            {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
13639          \ifdim\dimen@>#2\relax
13640            #2=\dimen@
13641          \fi
13642        }%
13643      }%
13644    }

```

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`.

```

13645    \newcommand*{\glsxtrComputeTreeIndent}[1]{%
13646      \glstreeindent=\glsxtrtreetopindent\relax
13647    }

```

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.

```

13648    \newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
13649      \ifcsundef{@glswidestname\romannumeral#1}%
13650      {%
13651        \settowidth{#3}{\glstreenamefmt{\@glswidestname\space}}%
13652      }%
13653      {%
13654        \settowidth{#3}{\glstreenamefmt{%
13655          \csname @glswidestname\romannumeral#1\endcsname\space}}%
13656      }%
13657    }

```

treeSetHangIndent Set `\hangindent` for top-level entries:

```

13658    \newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}

```

etSubHangIndent Set \hangindent for sub-entries:

```
13659 \newcommand*{\glxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}
```

Redefine alttree:

```
13660 \renewglossarystyle{alttree}{%
13661   \renewenvironment{theglossary}%
13662     {%
13663       \glxtralttreeInit
13664       \def\@gls@prevlevel{-1}%
13665       \mbox{}\par}%
13666     {\par}%
13667   \renewcommand*{\glossaryheader}{}%
13668   \renewcommand*{\glsgroupheading}[1]{}%
13669   \renewcommand{\glossentry}[2]{%
13670     \ifnum\@gls@prevlevel=0\relax
13671     \else
13672       \glxtrComputeTreeIndent{##1}%
13673     \fi
13674     \parindent\glstreeindent
13675     \glxtrAltTreeSetHangIndent
13676     \makebox[0pt][r]{%
13677       {%
13678         \glstreenamebox{\glstreeindent}%
13679         {%
13680           \glssentryitem{##1}%
13681           \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
13682         }%
13683       }%
13684       \glxtralttreeSymbolDescLocation{##1}{##2}%
13685       \def\@gls@prevlevel{0}%
13686     }
13687   \renewcommand{\subglossentry}[3]{%
13688     \ifnum##1=1\relax
13689       \glssubentryitem{##2}%
13690     \fi
13691     \ifnum\@gls@prevlevel=##1\relax
13692     \else
13693       \glxtrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
13694       \ifnum\@gls@prevlevel<##1\relax
13695         \setlength\glstreeindent\gls@tmplen
13696         \addtolength\glstreeindent\parindent
13697         \parindent\glstreeindent
13698       \else
13699         \ifnum\@gls@prevlevel=0\relax
13700           \glxtrComputeTreeIndent{##2}%
13701         \else
13702           \glxtrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
13703         \fi
13704         \addtolength\parindent{-\glstreeindent}%

```

```

13705         \setlength\glstreeindent\parindent
13706     \fi
13707 \fi
13708 \glxtrAltTreeSetSubHangIndent{##1}%
13709 \makebox[Opt][r]{\glstreenamebox{\gls@tmplen}{%
13710     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
13711 \glxtralttreeSubSymbolDescLocation{##1}{##2}{##3}%
13712 \def\@gls@prevlevel{##1}%
13713 }%
13714 \renewcommand*\{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
13715 }
13716 }%
13717 {%
13718 }

```

Redefine `almtreegroup` so that it discourages a break after group headings. Can't use `\@afterheading` here as it messes with the first item of the group.

```

13719 \ifdef{\@glsstyle@almtreegroup}
13720 {%
13721     \renewglossarystyle{almtreegroup}{%
13722         \setglossarystyle{almtree}%
13723         \renewcommand{\glsgroupheading}[1]{\par
13724             \def\@gls@prevlevel{-1}%
13725             \hangindentOpt\relax
13726             \parindentOpt\relax
13727             \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
13728         \nopagebreak\indexspace\nopagebreak
13729     }%
13730 }%
13731 }%
13732 {%
13733 }

```

Similarly for `almtreehypergroup`.

```

13734 \ifdef{\@glsstyle@almtreehypergroup}
13735 {%
13736     \renewglossarystyle{almtreehypergroup}{%
13737         \setglossarystyle{almtree}%
13738         \renewcommand*\{\glossaryheader}{%
13739             \par
13740             \def\@gls@prevlevel{-1}%
13741             \hangindentOpt\relax
13742             \parindentOpt\relax
13743             \glstreenavigationfmt{\glsnavigation}\par\indexspace
13744         }%
13745         \renewcommand*\{\glsgroupheading}[1]{%
13746             \par
13747             \def\@gls@prevlevel{-1}%
13748             \hangindentOpt\relax
13749             \parindentOpt\relax

```

```

13750      \glstreegroupheaderfmt
13751      {\glsnahypertarget{##1}{\glsgrouptitle{##1}}}\par
13752      \nopagebreak\indexspace\nopagebreak
13753  }%
13754  }
13755 }%
13756 {%
13757 }

```

2.9 Multicolumn Styles

Adjust mcolindexgroup to discourage page breaks after the group headings.

```

13758 \ifdef{\@glsstyle@mcolindexgroup}
13759 {%
13760   \renewglossarystyle{mcolindexgroup}{%
13761     \setglossarystyle{mcolindex}%
13762     \renewcommand*\{glsgroupheading}[1]{%
13763       \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
13764       \nopagebreak\indexspace\nobreak\@afterheading
13765     }%
13766   }
13767 }%
13768 {%
13769 }

```

Similarly for mcolindexhypergroup.

```

13770 \ifdef{\@glsstyle@mcolindexhypergroup}
13771 {%
13772   \renewglossarystyle{mcolindexhypergroup}{%
13773     \setglossarystyle{mcolindex}%
13774     \renewcommand*\{glossaryheader}{%
13775       \item\glstreenavigationfmt{\glsnavigation}%
13776       \indexspace
13777     }%
13778     \renewcommand*\{glsgroupheading}[1]{%
13779       \item\glstreegroupheaderfmt
13780       {\glsnahypertarget{##1}{\glsgrouptitle{##1}}}%
13781       \nopagebreak\indexspace\nobreak\@afterheading
13782     }%
13783   }
13784 }%
13785 {%
13786 }

```

Similarly for mcolindexspannav.

```

13787 \ifdef{\@glsstyle@mcolindexspannav}
13788 {%
13789   \renewglossarystyle{mcolindexspannav}{%
13790     \setglossarystyle{index}%

```

```

13791 \renewenvironment{theglossary}%
13792 {%
13793 \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]]%
13794 \setlength{\parindent}{0pt}%
13795 \setlength{\parskip}{0pt plus 0.3pt}%
13796 \let\item\glstreeitem}%
13797 {\end{multicols}}}%
13798 \renewcommand*{\glsgroupheading}[1]{%
13799 \item\glstreegroupheaderfmt
13800 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13801 \nopagebreak\indexspace\nobreak\@afterheading
13802 }%
13803 }
13804 }%
13805 {%
13806 }

```

Similarly for mcoltreegroup.

```

13807 \ifdef{\@glsstyle@mcoltreegroup}
13808 {%
13809 \renewglossarystyle{mcoltreegroup}{%
13810 \setglossarystyle{mcoltree}%
13811 \renewcommand{\glsgroupheading}[1]{\par
13812 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}}%
13813 \nopagebreak\indexspace\nobreak\@afterheading
13814 }%
13815 }
13816 }%
13817 {%
13818 }

```

Similarly for mcoltreehypergroup.

```

13819 \ifdef{\@glsstyle@mcoltreehypergroup}
13820 {%
13821 \renewglossarystyle{mcoltreehypergroup}{%
13822 \setglossarystyle{mcoltree}%
13823 \renewcommand*{\glossaryheader}{%
13824 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace
13825 }%
13826 \renewcommand*{\glsgroupheading}[1]{%
13827 \par\noindent
13828 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
13829 \nopagebreak\indexspace\nobreak\@afterheading
13830 }%
13831 }
13832 }%
13833 {%
13834 }

```

Similarly for mcoltreespannav.

```

13835 \ifdef{\@glsstyle@mcoltreespannav}

```

```

13836 {%
13837   \renewglossarystyle{mcoltreesspannav}{%
13838     \setglossarystyle{tree}%
13839     \renewenvironment{theglossary}%
13840     {%
13841       \begin{multicols}{\glsmcols}%
13842       [\noindent\glstreenavigationfmt{\glsnavigation}]%
13843       \setlength{\parindent}{0pt}%
13844       \setlength{\parskip}{0pt plus 0.3pt}%
13845     }%
13846     {\end{multicols}}}%
13847   \renewcommand*{\glsgroupheading}[1]{%
13848     \par\noindent
13849     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13850     \nopagebreak\indexspace\nobreak\@afterheading
13851   }%
13852 }
13853 }%
13854 {%
13855 }

```

Similarly for mcoltreenonamegroup.

```

13856 \ifdef{\@glsstyle@mcoltreenonamegroup}
13857 {%
13858   \renewglossarystyle{mcoltreenonamegroup}{%
13859     \setglossarystyle{mcoltreenoname}%
13860     \renewcommand*{\glsgroupheading}[1]{\par
13861       \noindent\glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%
13862     \nopagebreak\indexspace\nobreak\@afterheading
13863   }%
13864 }
13865 }%
13866 {%
13867 }

```

Similarly for mcoltreenonamehypergroup.

```

13868 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
13869 {%
13870   \renewglossarystyle{mcoltreenonamehypergroup}{%
13871     \setglossarystyle{mcoltreenoname}%
13872     \renewcommand*{\glossaryheader}{%
13873       \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
13874     \renewcommand*{\glsgroupheading}[1]{%
13875       \par\noindent
13876       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13877       \nopagebreak\indexspace\nobreak\@afterheading}%
13878   }
13879 }%
13880 {%
13881 }

```

Similarly for mcoltreenonamespannav.

```

13882 \ifdef{\@glsstyle@mcoltreenonamespannav}
13883 {%
13884   \renewglossarystyle{mcoltreenonamespannav}{%
13885     \setglossarystyle{treenoname}%
13886     \renewenvironment{theglossary}%
13887     {%
13888       \begin{multicols}{\glsmcols}%
13889       [\noindent\glstreenavigationfmt{\glsnavigation}}%
13890       \setlength{\parindent}{0pt}%
13891       \setlength{\parskip}{0pt plus 0.3pt}%
13892     }%
13893     {\end{multicols}}}%
13894   \renewcommand*{\glsgroupheading}[1]{%
13895     \par\noindent
13896     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13897     \nopagebreak\indexspace\nobreak\@afterheading}%
13898   }
13899 }%
13900 {%
13901 }

```

mcolalmtree needs adjusting so that it uses \glxtralttreeInit This doesn't use \mbox{\par which would unbalance the top of the columns.

```

13902 \ifdef{\@glsstyle@mcolalmtree}
13903 {%
13904   \renewglossarystyle{mcolalmtree}{%
13905     \setglossarystyle{almtree}%
13906     \renewenvironment{theglossary}%
13907     {%
13908       \glxtralttreeInit
13909       \def\@gls@prevlevel{-1}%
13910       \begin{multicols}{\glsmcols}%
13911     }%
13912     {\par\end{multicols}}}%
13913   }
13914 }%
13915 {%
13916 }

```

Redefine mcolalmtreegroup to discourage page breaks after the group headings.

```

13917 \ifdef{\@glsstyle@mcolalmtreegroup}
13918 {%
13919   \renewglossarystyle{mcolalmtreegroup}{%
13920     \setglossarystyle{mcolalmtree}%
13921     \renewcommand{\glsgroupheading}[1]{\par
13922       \def\@gls@prevlevel{-1}%
13923       \hangindent0pt\relax
13924       \parindent0pt\relax
13925       \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}}%

```

```

13926      \nopagebreak\indexspace\nopagebreak
13927    }%
13928  }
13929 }%
13930 {%
13931 }

```

Similarly for mcolalttreehypergroup.

```

13932 \ifdef{\@glsstyle@mcolalttreehypergroup}
13933 {%
13934   \renewglossarystyle{mcolalttreehypergroup}{%
13935     \setglossarystyle{mcolalttree}%
13936     \renewcommand*{\glossaryheader}{%
13937       \par
13938       \def\@gls@prevlevel{-1}%
13939       \hangindent0pt\relax
13940       \parindent0pt\relax
13941       \glstreenavigationfmt{\glsnavigation}%
13942       \par\indexspace
13943     }%
13944     \renewcommand*{\glsgroupheading}[1]{%
13945       \par
13946       \def\@gls@prevlevel{-1}%
13947       \hangindent0pt\relax
13948       \parindent0pt\relax
13949       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%
13950       \nopagebreak\indexspace\nopagebreak
13951     }%
13952   }
13953 }%
13954 {%
13955 }

```

Similarly for mcolalttreespannav.

```

13956 \ifdef{\@glsstyle@mcolalttreespannav}
13957 {%
13958   \renewglossarystyle{mcolalttreespannav}{%
13959     \setglossarystyle{alttree}%
13960     \renewenvironment{theglossary}%
13961     {%
13962       \glsextralmtreeInit
13963       \def\@gls@prevlevel{-1}%
13964       \begin{multicols}{\glsmcols}%
13965         [\noindent\glstreenavigationfmt{\glsnavigation}]%
13966     }%
13967     {\par\end{multicols}}%
13968     \renewcommand*{\glsgroupheading}[1]{%
13969       \par
13970       \def\@gls@prevlevel{-1}%
13971       \hangindent0pt\relax

```

```

13972      \parindent0pt\relax
13973      \glstreegroupheaderfmt{\glsnahypertarget{##1}{\glsgrouptitle{##1}}}%
13974      \nopagebreak\indexspace\nopagebreak
13975    }%
13976  }
13977}%
13978{%
13979}

Reset the default style
13980\ifx\@glossary@default@style\relax
13981\else
13982  \setglossarystyle{\@glxtr@current@style}
13983\fi

```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
13984 \NeedsTeXFormat{LaTeX2e}
13985 \ProvidesPackage{glossary-bookindex}[2018/04/09 v1.29 (NLCT)]
```

Load required packages.

```
13986 \RequirePackage{multicol}
13987 \RequirePackage{glossary-tree}
```

`trbookindexcols` Number of columns.

```
13988 \newcommand{\glstrbookindexcols}{2}
```

`trbookindexname` Format used for top-level entries. (Argument is the label.)

```
13989 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

`ookindexsubname` Format used for sub entries.

```
13990 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

`xsxtrprelocation` Provide in case glossaries-stylemods isn't loaded.

```
13991 \providecommand*{\glsxtrprelocation}{\space}
```

`ndexprelocation` 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.

```
13992 \newcommand*{\glstrbookindexprelocation}[1]{%
13993   \glstrifhasfield{location}{#1}%
13994   {,\glsxtrprelocation}%
13995   {\glsxtrprelocation}%
13996 }
```

`xsubprelocation` Separator used before location list for sub-entries.

```
13997 \newcommand*{\glstrbookindexsubprelocation}[1]{%
13998   \glsxtrbookindexprelocation{#1}%
13999 }
```

`xparentchildsep` Separator used between top-level parent and child entry.

```
14000 \newcommand{\glstrbookindexparentchildsep}{\nopagebreak}
```

`rentsubchildsep` Separator used between sub-level parent and child entry.

```
14001 \newcommand{\glstrbookindexparentsubchildsep}{\glstrbookindexparentchildsep}
```

`bookindexbetween` Between two top-level entries identified by the labels in the arguments.

```
14002 \newcommand{\glxstrbookindexbetween}[2] {}
```

`indexsubbetween` Between two level 1 entries identified by the labels in the arguments.

```
14003 \newcommand{\glxstrbookindexsubbetween}[2] {}
```

`exsubsubbetween` Between two level 2 entries identified by the labels in the arguments.

```
14004 \newcommand{\glxstrbookindexsubsubbetween}[2] {}
```

`indexatendgroup` At the end of a letter group. The argument is the index of the last top-level entry.

```
14005 \newcommand{\glxstrbookindexatendgroup}[1] {}
```

`exsubatendgroup` At the end of a letter group. The argument is the index of the last level 1 entry.

```
14006 \newcommand{\glxstrbookindexsubatendgroup}[1] {}
```

`subsubatendgroup` At the end of a letter group. The argument is the index of the last level 2 entry.

```
14007 \newcommand{\glxstrbookindexsubsubatendgroup}[1] {}
```

`kindexgroupskip` Group separator.

```
14008 \newcommand{\glxstrbookindexgroupskip}{\ifglsnogroupskip\else\indexspace\fi}
```

Format group title.

`indexformatheader` Group separator.

```
14009 \newcommand*{\glxstrbookindexformatheader}[1] {%
14010   \par{\centering\glstreegroupheaderfmt{#1}\par}%
14011 }
```

`bookindexbookmark` Book mark group heading if supported.

```
14012 \ifdef\pdfbookmark
14013 {%
14014   \newcommand*{\glxstrbookindexbookmark}[2] {%
14015     \ifdefstring{\@@glossarysec}{chapter}%
14016     {\pdfbookmark[1]{#1}{#2}}%
14017     {\pdfbookmark[2]{#1}{#2}}%
14018   }
14019 }
14020 {%
14021   \newcommand*{\glxstrbookindexbookmark}[2] {}
14022 }
```

`kindexcolspread`

```
14023 \newcommand*{\glxstrbookindexcolspread} {}
```

`indexmulticolenv`

```
14024 \newcommand*{\glxstrbookindexmulticolenv}{multicols}
```

Define the style.

```

14025 \newglossarystyle{bookindex}{%
14026   \setglossarystyle{index}%
14027   \renewenvironment{theglossary}%
14028   {%
14029     \ifdefempty{glstrbookindexcolspread}
14030     {%
14031       \expandafter\begin\expandafter{\glstrbookindexmulticolseenv}%
14032       {\glstrbookindexcols}%
14033     }%
14034     {%
14035       \expandafter\begin\expandafter{\glstrbookindexmulticolseenv}%
14036       {\glstrbookindexcols}[\glstrbookindexcolspread]%
14037     }%
14038     \setlength{\parindent}{0pt}%
14039     \setlength{\parskip}{0pt plus 0.3pt}%
14040     \let\@glstrbookindex@sep\glstrbookindexparentchildsep
14041     \let\@glstrbookindex@subsep\glstrbookindexparentschildsep
14042     \let\@glstrbookindex@between\@gobble
14043     \let\@glstrbookindex@subbetween\@gobble
14044     \let\@glstrbookindex@subsubbetween\@gobble
14045     \let\@glstrbookindex@atendgroup\relax
14046     \let\@glstrbookindex@subatendgroup\relax
14047     \let\@glstrbookindex@subsubatendgroup\relax
14048     \let\@glstrbookindex@groupskip\relax
14049   }%
14050   {%

```

Do end group hooks.

```

14051     \@glstrbookindex@subsubatendgroup
14052     \@glstrbookindex@subatendgroup
14053     \@glstrbookindex@atendgroup

```

End multicol environment.

```

14054     \expandafter\end\expandafter{\glstrbookindexmulticolseenv}%
14055   }%

```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```

14056   \renewcommand*{\glossaryheader}{\raggedright}%

```

Top level entry format.

```

14057   \renewcommand*{\glossentry}[2]{%

```

Do separator.

```

14058     \@glstrbookindex@between{##1}%

```

Update separators.

```

14059     \let\@glstrbookindex@sep\glstrbookindexparentchildsep
14060     \let\@glstrbookindex@subsep\glstrbookindexparentschildsep
14061     \let\@glstrbookindex@subbetween\@gobble
14062     \let\@glstrbookindex@subsubbetween\@gobble
14063     \edef\@glstrbookindex@between{%

```

```

14064      \noexpand\glxstrbookindexbetween{##1}%
14065  }%
14066  \edef\@glxstr@bookindex@atendgroup{%
14067      \noexpand\glxstrbookindexatendgroup{##1}%
14068  }%
14069  \let\@glxstr@bookindex@subatendgroup\relax
14070  \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

14071  \glstreeitem
14072      \glstryitem{##1}%
14073      \glstarget{##1}{\glxstrbookindexname{##1}}%
14074  \glxstrbookindexprelocation{##1}##2%
14075  }%
14076  \renewcommand{\subglossentry}[3]{%
14077      \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```

14078      \glstreeitem
14079      \or

```

Level 1.

```

14080      \@glxstr@bookindex@sep
14081      \@glxstr@bookindex@subbetween{##2}%
14082      \let\@glxstr@bookindex@sep\relax

```

Update separators.

```

14083      \let\@glxstr@bookindex@subsubbetween\@gobble
14084      \let\@glxstr@bookindex@subsep\glxstrbookindexparentschildsep
14085      \edef\@glxstr@bookindex@subbetween{%
14086          \noexpand\glxstrbookindexsubbetween{##2}%
14087      }%
14088      \edef\@glxstr@bookindex@atsubendgroup{%
14089          \noexpand\glxstrbookindexatsubendgroup{##1}%
14090      }%

```

Start sub-item.

```

14091      \glstreesubitem
14092      \glssubentryitem{##2}%
14093      \else

```

All other levels.

```

14094      \@glxstr@bookindex@subsep
14095      \@glxstr@bookindex@subsubbetween{##2}%

```

Update separators.

```

14096      \let\@glxstr@bookindex@subsep\relax
14097      \edef\@glxstr@bookindex@subsubbetween{%
14098          \noexpand\glxstrbookindexsubsubbetween{##2}%
14099      }%
14100      \edef\@glxstr@bookindex@atsubsubendgroup{%
14101          \noexpand\glxstrbookindexatsubsubendgroup{##1}%
14102      }%

```

Start sub-sub-item.

```
14103 \glstreesubsubitem
14104 \fi
```

Format entry.

```
14105 \glstarget{##2}{\glxtrbookindexsubname{##2}}%
14106 \glxtrbookindexsubprelocation{##2}##3%
14107 }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
14108 \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
14109 \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
14110 \@glxtr@bookindex@subsubatendgroup
14111 \@glxtr@bookindex@subatendgroup
14112 \@glxtr@bookindex@atendgroup
14113 \@glxtr@bookindexgroupskip
```

Update separators.

```
14114 \let\@glxtr@bookindexgroupskip\glxtrbookindexgroupskip
14115 \let\@glxtr@bookindex@between\@gobble
14116 \let\@glxtr@bookindex@atendgroup\relax
14117 \let\@glxtr@bookindex@subatendgroup\relax
14118 \let\@glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
14119 \glxtrgetgrouptitle{##1}{\thisgrptitle}%
```

Do the PDF bookmark if supported.

```
14120 \glxtrbookindexbookmark{\thisgrptitle}{index.##1}%
```

Format the group title.

```
14121 \glxtrbookindexformatheader{\thisgrptitle}%
14122 \nopagebreak\indexspace\nopagebreak\@afterheading
14123 }%
14124 }
```

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.

```
14125 \newcommand{\glxtrbookindexthepage}{%
14126 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
14127 }
```

bookindexmarkentry Writes entry information to the .aux file. The argument is the entry label.

```
14128 \newcommand*\glxtrbookindexmarkentry}[1]{%
14129   \protected@write\@auxout
14130   {\let\glxtrbookindexthepage\relax}%
14131   {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
14132 }
```

etbookindexmark

```
14133 \newcommand*\glxtr@setbookindexmark}[2]{%
14134   \ifcsundef\glxtr@idxfirstmark@#1}%
14135   {\csgdef\glxtr@idxfirstmark@#1}{#2}}%
14136   {}%
14137   \csgdef\glxtr@idxlastmark@#1}{#2}%
14138 }
```

indexfirstmarkfmt

```
14139 \newcommand*\glxtrbookindexfirstmarkfmt}[1]{%
14140   \glxentryname{#1}%
14141 }
```

indexfirstmark

```
14142 \newcommand*\glxtrbookindexfirstmark}{%
14143   \letcs\glxtr@label\glxtr@idxfirstmark@\glxtrbookindexthepage}%
14144   \ifdef\glxtr@label
14145   {\glxtrbookindexfirstmarkfmt\glxtr@label}}%
14146   {}%
14147 }
```

indexlastmarkfmt

```
14148 \newcommand*\glxtrbookindexlastmarkfmt}[1]{%
14149   \glxentryname{#1}%
14150 }
```

okindexlastmark

```
14151 \newcommand*\glxtrbookindexlastmark}{%
14152   \letcs\glxtr@label\glxtr@idxlastmark@\glxtrbookindexthepage}%
14153   \ifdef\glxtr@label
14154   {\glxtrbookindexlastmarkfmt\glxtr@label}}%
14155   {}%
14156 }
```

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	67
General: Initial experimental release	5	\@Glsymbolplural@: added	
0.2 (2015-11-30)		redefinition	68
\Glsfmtshort: new	309	\@Glstext@: added redefinition	63
\glsfmtshort: new	308	\@Glsuseri@: added redefinition	68
\Glsfmtshortpl: new	309	\@Glsuserii@: added redefinition	68
\glsfmtshortpl: new	308	\@Glsuseriii@: added redefinition	69
short: switched inline full form to short		\@Glsuseriv@: added redefinition	69
(long)	212	\@Glsuserv@: added redefinition	69
0.3 (2015-12-02)		\@Glsuservi@: added redefinition	70
\@ACRlong: added redefinition	73	\@acrlong: added redefinition	72
\@ACRlongpl: added redefinition	74	\@acrlongpl: added redefinition	73
\@ACRshort: added redefinition	71	\@acrshort: added redefinition	70
\@ACRshortpl: added redefinition	72	\@acrshortpl: added redefinition	71
\@Acrlong: added redefinition	72	\@gls@field@link: added optional	
\@Acrlongpl: added redefinition	73	argument	56
\@Acrshort: added redefinition	70	\@glsdescplural@: added redefinition	66
\@Acrshortpl: added redefinition	71	\@glsfirst@: added redefinition	63
\@GLSdesc@: added redefinition	66	\@glsfirstplural@: added redefinition	65
\@GLSdescplural@: added redefinition	67	\@glsplural@: added redefinition	64
\@GLSfirst@: added redefinition	64	\@glssymbolplural@: added	
\@GLSfirstplural@: added redefinition	65	redefinition	67
\@GLSname@: added redefinition	66	\@glsxtr@defaultnoglossarywarning:	
\@GLSplural@: added redefinition	65	new	124
\@GLSSymbol@: added redefinition	67	\@glsxtr@field@linkdefs: new	62
\@GLSSymbolplural@: added		\@glsxtr@insertdots: new	181
redefinition	68	\@print@glossary: added redefinition	120
\@GLStext@: added redefinition	63	\glsabbrvdefaultfont: renamed from	
\@GLSuseri@: added redefinition	68	\abbrvdefaultfont	186
\@GLSuserii@: added redefinition	69	\glsaccessdesc: new	149
\@GLSuseriii@: added redefinition	69	\glsaccessdescplural: new	149
\@GLSuseriv@: added redefinition	69	\glsaccessfirst: new	146
\@GLSuserv@: added redefinition	69	\glsaccessfirstplural: new	147
\@GLSuservi@: added redefinition	70	\Glsaccesslong: new	151
\@Glsdesc@: added redefinition	66	\glsaccesslong: new	151
\@Glsdescplural@: added redefinition	67	\glsaccessname: new	144
\@Glsfirst@: added redefinition	64	\glsaccessplural: new	146
\@Glsfirstplural@: added redefinition	65	\Glsaccessshort: new	150
\@Glsname@: added redefinition	66	\glsaccessshort: new	150
\@Glsplural@: added redefinition	64	\Glsaccessshortpl: new	150

\glsaccesssshortpl: new	150	\cGLSpl: new	97
\glsaccessssymbol: new	147	\cGLSpl@: new	98
\glsaccessssymbolplural: new	148	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	145	new	93
\glstentryfmt: added check for short ..	56	\cGLS: new	97
\glslongpltok: new	181	\cGLSformat: new	97
\glsshortpltok: new	181	\cGLSpl: new	97
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	98
name in \setkeys	182	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	15
for plural	178	\glsenableentrycount: new	93
\GLSxtrlongpl: new	196	\glsfirstabbrvdefaultfont: new ..	186
\Glsxtrlongpl: new	195	\glsfirstlongdefaultfont: new ...	187
\glsxtrlongpl: new	195	\Glsfmtfirst: new	311
\glsxtrNoGlossaryWarning: new	21	\glsfmtfirst: new	311
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	312
new	177	\glsfmtfirstpl: new	311
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	311
new	178	\glsfmtplural: new	310
\glsxtrpostlinkendsentence: new ..	177	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	194	\Glsxtrtitleshort	309
\Glsxtrshortpl: new	194	renamed from \glstentryfmtshort ..	309
\glsxtrshortpl: new	193	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort	308
\glslabeltok	206	renamed from \glstentryfmtshort ..	308
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	204	\Glsxtrtitleshortpl	309
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl	309
redefinition of \acronymtype	17	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl	308
\Glsxtrshort	309	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl	308
\glsxtrshort	308	\Glsfmttext: new	310
\Glsfmtshortpl: changed to use		\glsfmttext: new	310
\glsxtrshortpl	309	\glshasattribute: new	156
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	156
\glsxtrshortpl	308	\glsxtremsuffix: new	248
\glsxtrifemptyglossary: new	27	\GlsXtrEnableEntryCounting: new ..	92
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	95
argument	160	\glsxtrscfont: new	219
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	220
argument	159	\glsxtrsmfont: new	234
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	234
default type to \acronymtype	106	short-em: new	255
\newterm: fixed name argument	159	short-em-desc: new	256
0.5 (2015-12-07)		short-em-footnote: new	266
\@cGLS: new	97	short-em-long: new	251
\@cGLS@: new	97	short-em-long-desc: new	253

short-em-postfootnote: new	267	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	230	attribute	299
short-sc-postfootnote: new	232	\Glsxtrheadtext: now uses headuc	
short-sm: new	237	attribute	302
short-sm-desc: new	239	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	244	attribute	301
short-sm-long: new	236	short-em-footnote: switch off regular	
short-sm-long-desc: new	237	attribute if set	266
short-sm-postfootnote: new	246	short-long: switch off regular attribute	
long-noshort-em: new	258	if set	205
long-noshort-em-desc: new	262	short-long-desc: switch off regular	
long-noshort-sm: new	241	attribute if set	207
long-noshort-sm-desc: new	243	short-sc-footnote: switch off regular	
long-short-em: new	248	attribute if set	230
long-short-em-desc: new	249	short-sm-footnote: switch off regular	
long-short-sm: new	234	attribute if set	244
long-short-sm-desc: new	235	long-short: switch off regular attribute	
0.5.1 (2015-12-02)		if set	203
\Glsaccessstext: new	145	long-short-desc: switch off regular	
0.5.1 (2015-12-07)		attribute if set	205
\@glxtr@doaccsupp: new	20	long-short-sc-desc: switch off regular	
General: removed \ifglxtruseuchead	299	attribute if set	221
\Glsaccessdesc: new	149	footnote: switch off regular attribute if	
\Glsaccessdescplural: new	149	set	208
\Glsaccessfirst: new	146	postfootnote: switch off regular	
\Glsaccessfirstplural: new	147	attribute if set	209
\Glsaccessname: new	145	0.5.2 (2015-12-08)	
\Glsaccessplural: new	146	\@GLSdesc@: added accessibility support	66
\Glsaccesssymbol: new	148	\@GLSdescplural@: added accessibility	
\Glsaccesssymbolplural: new	148	support	67
\glxtrheadfirst: now uses headuc		\@GLSfirst@: added accessibility	
attribute	303	support	64
\glxtrheadfirst: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	303	support	65
\Glsxtrheadfirstplural: now uses		\@GLSname@: added accessibility support	66
headuc attribute	304	\@GLSplural@: added accessibility	
\glxtrheadfirstplural: now uses		support	65
headuc attribute	304	\@GLSsymbol@: added accessibility	
\Glsxtrheadplural: now uses headuc		support	67
attribute	303	\@GLSsymbolplural@: added	
\glxtrheadplural: now uses headuc		accessibility support	68
attribute	302	\@GLStext@: added accessibility support	63
\Glsxtrheadshort: now uses headuc		\@GLSdesc@: added accessibility support	66
attribute	300	\@GLSdescplural@: added accessibility	
\glxtrheadshort: now uses headuc		support	67
attribute	299	\@GLSfirst@: added accessibility	
\Glsxtrheadshortpl: now uses headuc		support	64
attribute	300	\@GLSfirstplural@: added accessibility	
		support	65

\@Glsname@: add accessibility support ..	66	\GLSaccesssymbolplural: new ..	148, 153
\@Glsplural@: added accessibility support	64	\GLSaccessstext: new	145, 152
\@Glsymbol@: added accessibility support	67	\glsentryfmt: moved	
\@Glsymbolplural@: added accessibility support	68	\glssetabbrvfmt from	
\@Glstext@: added accessibility support	63	\glsxtrabbrvfmt to here	56
\@glsdesc@: added accessibility support	66	\GlsXtrEnableInitialTagging: new	174
\@glsdescplural@: added accessibility support	66	\glsxtrfieldtitlecase: new	160
\@glsfirst@: added accessibility support	63	\GlsXtrFormatLocationList: new ...	54
\@glsfirstplural@: added accessibility support	65	\glsxtrnewabbrevpresetkeyhook:	
\@Glsname@: added accessibility support	66	new	185
\@Glsplural@: added accessibility support	64	\glsxtrtagfont: new	175
\@Glsymbol@: added accessibility support	67	\KV@printgloss@nonumberlist: added	55
\@Glsymbolplural@: added accessibility support	67	\mfu@checkword@do: added	174
\@Glstext@: added accessibility support	63	\setabbreviationstyle: added check	
\@glsxtr@activate@initialtagging:		for post-definition style switch	199
new	175	0.5.3 (2015-12-09)	
\@glsxtr@do@titlecaps@warn: new ..	175	\@glsxtr@autoindex@at: new	171
\@glsxtr@tag: new	175	\@glsxtr@autoindex@encap: new ...	171
General: fixed typo in glossaries-accsupp		\@glsxtr@autoindex@esc: new	172
and tidied up code to use just one		\@glsxtr@autoindex@level: new ...	172
\@ifpackageloaded	144	\@glsxtr@autoindex@setname: new ..	170
removed \glsxtrabbrvfmt	197	\@glsxtr@doabbreviationsdef: new ..	17
\glossaryentrynumbers: added	53	General: removed	
\Glossentrydesc: added	173	\GlsXtrNoGlsWarningNoAutoMakeMain	
\Glossentryname: added	165	123
\Glossentrysymbol: added	173	\glsdescwidth: added	53
\Glsaccessdesc: new	149, 154	\glspagelistwidth: added	53
\Glsaccessdescplural: new ...	149, 154	\glsxtrdoautoindexname: new	169
\Glsaccessfirst: new	147, 153	\glsxtrpostnamehook: new	166
\Glsaccessfirstplural: new ..	147, 153	\if@glsxtr@format@override: new ..	168
\Glsaccesslong: new	151, 155	\ProvidesGlossariesExtraLang: new	315
\Glsaccesslongpl: new	152, 155	\RequireGlossariesExtraLang: new	314
\Glsaccesslongpl: new	151	0.5.4 (2015-12-15)	
\Glsaccesslongpl: new	151	\@newglossaryentry@defunitcounters:	
\Glsaccessname: new	145, 152	new	98
\Glsaccessplural: new	146, 152	\@GLSxtr@p@acrlong@: new	86
\Glsaccessshort: new	150, 154	\@GLSxtr@p@acrlongpl@: new	86
\Glsaccessshortpl: new	151, 154	\@GLSxtr@p@acrshort@: new	85
\Glsaccesssymbol: new	148, 153	\@GLSxtr@p@acrshortpl@: new	85
		\@GLSxtr@p@long@: new	85
		\@GLSxtr@p@longpl@: new	85
		\@GLSxtr@p@plural@: new	84
		\@GLSxtr@p@short@: new	84
		\@GLSxtr@p@shortpl@: new	84
		\@GLSxtr@p@text@: new	83
		\@GlsXtrEnableOnTheFly: new	49
		\@Glsxtr: new	50
		\@Glsxtr@p@acrlong@: new	86
		\@Glsxtr@p@acrlongpl@: new	86

\@Glsxtr@p@acrshort@: new	85	\glsenableentryunitcount: new	101
\@Glsxtr@p@acrshortpl@: new	85	\glshasattribute: added check for	
\@Glsxtr@p@long@: new	85	entry's existence	156
\@Glsxtr@p@longpl@: new	85	\glusifattribute: added check for	
\@Glsxtr@p@plural@: new	84	entry's existence	157
\@Glsxtr@p@short@: new	84	\glspostlinkhook: added existence	
\@Glsxtr@p@shortpl@: new	84	check	177
\@Glsxtr@p@text@: new	83	\Glsxtr: new	50
\@Glsxtrpl: new	51	\glxtr: new	49
\@alt@glshyp@opt: new	80	\glxtrcat: new	49
\@glscalt@hyp@opt: new	79	\glxtrdowrglossaryhook: new	79
\@glscalt@hyp@opt@char: new	80	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new	80	new	104
\@glsc@increment@currunitcount:		\GlsXtrEnableOnTheFly: new	48
new	99	\Glsxtrpl: new	50
\@glsc@local@increment@currunitcount:		\glxtrpl: new	50
new	100	\glxtrpostlocalreset: new	92
\@glsc@setdefault@glslink@opts:		\glxtrpostlocalunset: new	92
new	77	\glxtrpostreset: new	92
\@glsxtr: new	49	\glxtrpostunset: new	91
\@glsxtr@addunitcounter: new	99	\glxtrprotectlinks: new	83
\@glsxtr@currunitcount: new	100	\GlsXtrSetAltModifier: new	80
\@glsxtr@ifunitcounter: new	99	\GlsXtrSetDefaultGlsOpts: new	78
\@glsxtr@p@acrlong@: new	85	\glxtrstarflywarn: new	49
\@glsxtr@p@acrlongpl@: new	86	\GlsXtrWarning: new	51
\@glsxtr@p@acrshort@: new	85	\MakeAcronymsAbbreviations: now	
\@glsxtr@p@acrshortpl@: new	85	disables \setacronymstyle	106
\@glsxtr@p@long@: new	85	1.0 (2016-01-24)	
\@glsxtr@p@longpl@: new	85	\@glsxtr@autoindexcrossrefs: new	15
\@glsxtr@p@plural@: new	83	\@glsxtr@idx@displaynumberlist:	
\@glsxtr@p@short@: new	84	new	114
\@glsxtr@p@shortpl@: new	84	\@glsxtr@idx@entrynumberlist: new	116
\@glsxtr@p@text@: new	83	\@glsxtr@noidx@displaynumberlist:	
\@glsxtr@prevunitcount: new	100	new	114
\@glsxtr@setentryunitcountunsetattr:		\@glsxtr@noidx@entrynumberlist:	
new	104	new	115
\@glsxtr@unitcountlist: new	99	\@glsxtr@noidx@numberlistloop:	
\@glsxtrpl: new	50	new	115
\@newglossaryentryposthook: added		\@glsxtr@reg@glosslist: new	107
empty see value if not set and added		\makeglossaries: new	107
'see' to field key map	40	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new	49	\glxtrdiscardperiod: added check	
\cGlsformat: added	98	for first use	178
\cGlsformat: added	98	short-desc: fixed typo in	
\cGlsplformat: added	98	\glxtrinlinfullformat and	
\cGlsplformat: added	98	added missing second argument	213
\glstdisablehyper: added	82	1.02 (2016-04-25)	
\glstdohyperlink: added	81	\@glsxtr@current@style: new	52
\glstdonohyperlink: added	82	\Glsfmtfull: new	313

\glsfmtfull: new	313	\@GLSdescplural@: set abbreviation and regular format	67
\Glsfmtfullpl: new	314	\@GLSfirst@: set abbreviation format ..	64
\glsfmtfullpl: new	314	\@GLSfirstplural@: set abbreviation and regular format	65
\Glsfmtlong: new	312	\@GLSname@: set abbreviation and regular format	66
\glsfmtlong: new	312	\@GLSplural@: set abbreviation and regular format	65
\Glsfmtlongpl: new	313	\@GLSsymbol@: set regular format	67
\glsfmtlongpl: new	313	\@GLSsymbolplural@: set regular format	68
\Glsxtrheadfull: new	307	\@GLStext@: set abbreviation and regular format	63
\glsxtrheadfull: new	306	\@GLSuseri@: set regular format	68
\Glsxtrheadfullpl: new	308	\@GLSuserii@: set regular format	69
\glsxtrheadfullpl: new	307	\@GLSuseriii@: set regular format	69
\Glsxtrheadlong: new	305	\@GLSuseriv@: set regular format	69
\glsxtrheadlong: new	305	\@GLSuservi@: set regular format	70
\Glsxtrheadlongpl: new	306	\@GLSdesc@: set abbreviation and regular format	66
\glsxtrheadlongpl: new	305	\@GLSdescplural@: set abbreviation and regular format	67
\Glsxtrtitlefull: new	307	\@GLSfirst@: set abbreviation and regular format	64
\glsxtrtitlefull: new	307	\@GLSfirstplural@: set abbreviation and regular format	65
\Glsxtrtitlefullpl: new	308	\@GLSname@: set abbreviation and regular format	66
\glsxtrtitlefullpl: new	307	\@GLSplural@: set abbreviation and regular format	64
\Glsxtrtitlelong: new	306	\@GLSsymbol@: set regular format	67
\glsxtrtitlelong: new	305	\@GLSsymbolplural@: set regular format	68
\Glsxtrtitlelongpl: new	306	\@GLStext@: set abbreviation and regular format	63
\glsxtrtitlelongpl: new	305	\@GLSuseri@: set regular format	68
\ifglxtrinsertinside: new	203	\@GLSuserii@: set regular format	68
postfootnote: added redef of		\@GLSuseriii@: set regular format	69
\glsxtrsetupfulldefs	210	\@GLSuseriv@: set regular format	69
stylemods: new	21	\@GLSuservi@: set regular format	70
1.03 (2016-04-27)		\@GLSdesc@: set abbreviation and regular format	66
\@GLSfirstplural@: bug fix: misspelt cs name	65	\@GLSdescplural@: set abbreviation and regular format	64
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	65	\@GLSsymbol@: set regular format	67
\@GLSfirstplural@: bug fix: misspelt cs name	65	\@GLSsymbolplural@: set regular format	68
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	64	\@GLStext@: set abbreviation and regular format	63
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	64	\@GLSuseri@: set regular format	68
\glsxtrtitlelongpl: bug fix: changed \glsxtrlong to \glsxtrlongpl ..	305	\@GLSuserii@: set regular format	68
\glsxtrtitleshortpl: bug fix: changed \glsxtrshort to \glsxtrshortpl	300	\@GLSuseriii@: set regular format	69
1.04 (2015-04-30)		\@GLSuseriv@: set regular format	69
short-em-footnote: renamed from “footnote-em”	266	\@GLSuservi@: set regular format	70
1.04 (2016-05-02)		\@gls@preglossaryhook: added check for entry's existence	175
\@glsxtrpostloctag: new	55	\@GLSdesc@: set abbreviation and regular format	66
\@GLSdesc@: set abbreviation and regular format	66	\@GLSdescplural@: set abbreviation and regular format	66
		\@GLSfirst@: set abbreviation and regular format	63

\@glsfirstplural@: set abbreviation and regular format	65	\glsxtruserparen: new	269
\@glsname@: set abbreviation and regular format	66	\glsxtrusersuffix: new	270
\@glsplural@: set abbreviation and regular format	64	\GlsXtrWarnDeprecatedAbbrStyle: new	202
\@glsymbol@: set regular format	67	short-em-long-em: new	253
\@glsymbolplural@: set regular format	67	short-em-long-em-desc: new	254
\@glstext@: set abbreviation and regular format	63	short-em-nolong: new	256
\@glsxtr@deprecated@abbrstyle: new	201	short-em-nolong-desc: new	258
\@glsxtr@do@style: new	22	short-em-postfootnote: renamed from “postfootnote-em”	267
\@glsxtr@dolocktag: new	55	short-footnote: new	209
\@glsxtr@idx@entrynumberlist: switched from \let to \newcommand	116	short-long-user: new	277
\@glsxtr@pagetag: new	55	short-long-user-desc: new	278
\@glsxtr@pagetag: new	55	short-nolong: new	213
\@glsxtr@prelocktag: new	55	short-nolong-desc: new	214
\@glsxtr@prelocktag: new	54, 55	short-postfootnote: new	211
\glossentrydesc: added glossdescfont attribute check	161	short-sc-footnote: renamed from “footnote-sc”	230
\Glossentryname: added glossnamefont attribute check	165	short-sc-nolong: new	224
\glossentryname: added glossnamefont attribute check	163	short-sc-nolong-desc: new	226
moved post name hook inside condition	165	short-sc-postfootnote: renamed from “postfootnote-sc”	232
\glsabbrvmfont: new	248	short-sm-footnote: renamed from “footnote-sm”	244
\glsabbrvuserfont: new	270	short-sm-nolong: new	239
\glsfirstabbrvmfont: new	248	short-sm-nolong-desc: new	240
\glsfirstabbrvuserfont: new	270	short-sm-postfootnote: renamed from “postfootnote-sm”	246
\glsfirstlongemfont: new	248	\letabbreviationstyle: new	201
\glsfirstlonguserfont: new	270	\newabbreviationstyle: bug fix: corrected test for existence	200
\glsifnotregularcategory: new ...	157	long-em-noshort-em: new	260
\glslongdefaultfont: new	186	long-em-noshort-em-desc: new	264
\glslongemfont: new	248	long-em-short-em: new	250
\glslongfont: new	186	long-em-short-em-desc: new	251
\glslonguserfont: new	270	long-noshort: new	219
\glsxtrassignfieldfont: new	62	long-noshort-desc: new	218
\GlsXtrEnablePreLocationTag: new .	54	long-noshort-em: renamed from “long-em”	258
\glsxtrfirstscfont: new	219	long-noshort-em-desc: renamed from “long-desc-em”	262
\glsxtrfirstsmfont: new	234	long-noshort-sc: renamed from “long-sc”	227
\glsxtrlongshortdescsort: new ...	204	long-noshort-sc-desc: renamed from “long-desc-sc”	228
\glsxtrpostnamehook: added category check	166	long-noshort-sm: renamed from “long-sm”	241
\glsxtrregularfont: new	56	long-noshort-sm-desc: renamed from \long-desc-sm	243
\glsxtruserfield: new	269		

long-short-user: new	270	docdef option changed to choice	14
long-short-user-desc: new	276	\glxtr@usesee: new	41
\renewabbreviationstyle: new	201	\glxtrusesee: new	41
style: new	22	\glxtruseseeformat: new	41
1.05 (2016-06-10)		\if@glxtrdocdefrestricted: new ..	15
\eglssetwidest: new	368	1.07 (2016-08-15)	
\glFindWidestAnyName: new	371	\@@glxtrp: new	86
\glFindWidestAnyNameLocation:		\@GLSfirst@: added check for	
new	376	nohyperfirst attribute	64
\glFindWidestAnyNameSymbol: new	374	\@GLSfirstplural@: added check for	
\glFindWidestAnyNameSymbolLocation:		nohyperfirst attribute	65
new	375	\@GLSxtrp: new	87
\glFindWidestLevelTwo: new	372	\@Glsfirst@: added check for	
\glFindWidestUsedAnyName: new ..	371	nohyperfirst attribute	64
\glFindWidestUsedAnyNameLocation:		\@Glsfirstplural@: added check for	
new	376	nohyperfirst attribute	65
\glFindWidestUsedAnyNameSymbol:		\@Glsxtrp: new	87
new	373	\@gls@preglossaryhook: added	
\glFindWidestUsedAnyNameSymbolLocation:		\glossxtrsetpopts	176
new	374	\@glsfirst@: added check for	
\glFindWidestUsedLevelTwo: new .	371	nohyperfirst attribute	64
\glFindWidestUsedTopLevelName:		\@glsfirstplural@: added check for	
new	370	nohyperfirst attribute	65
\glfirstlongfootnotefont: new ..	207	\@glxtrinmark: new	297
\glsgetwidestname: new	370	\@glxtrnotinmark: new	297
\glsgetwidestsubname: new	370	\@glxtrp: new	86
\glslongfootnotefont: new	207	\@glxtrp@opt: new	86
\glxtrAltTreeIndent: new	368	\glossxtrsetpopts: new	86
\glxtralttreeInit: new	368	\glsp: new	89
\glxtrAltTreePar: new	368	\glsp: new	89
\glxtrAltTreeSetHangIndent: new	377	\glxtr@entry@p: new	87
\glxtrAltTreeSetSubHangIndent:		\glxtrabbrvfootnote: new	207
new	378	\glxtrchecknohyperfirst: new	63
\glxtralttreeSubSymbolDescLocation:		\glxtrfieldtitlecasecs: new	161
new	368	\glxtrifinmark: new	296
\glxtralttreeSymbolDescLocation:		\GLSxtrp: new	90
new	367	\Glsxtrp: new	89
\glxtrComputeTreeIndent: new ...	377	\glxtrp: new	88
\glxtrComputeTreeSubIndent: new	377	\glxtrsetpopts: new	86
\glxtrtreetopindent: new	368	short-long-desc: added text key	207
short-em-long: fixed incorrect font used		fixed misspelling of \glabbrvfont in	
by long form	252	plural key	207
\xglsetwidest: new	369	long-short-desc: added missing text	
1.06 (2016-06-18)		key	204
\@glsoifexistsorwarn: new	15	fixed misspelling of \glabbrvfont .	205
\@glxtr@docdefval: new	14	footnote: changed first forms to use	
\@glxtr@usesee: new	41	\glfirstlongfootnotefont ...	208
General: disabled docdef key at the start		postfootnote: removed \footnote	
of the document	27	from first keys	209

switched from \glsfirstlongfont to \glsfirstlongfootnotefont ...	210	1.10 (2016-12-17)	\@GLSp1@: fixed bug caused by typo in command name	58
\RestoreAcronyms: modified \@gls@link@checkfirsthyper to set \glsxtrifwasfirstuse	107	1.11 (2017-01-19)	\@glsxtr@do@redef@for@gl@sentries: new	6
1.08 (2016-12-13)			\@glsxtr@noidx@do: new	134
\@glsxtr@record: new	8		\@glsxtr@redef@for@gl@sentries: new ..	6
\@GLS@: added \@glsxtr@record	58		\@glsxtr@shortcutsval: new	19
\@GLSp1@: added \@glsxtr@record ...	58		\@glsxtr@unsrt@getgroup@title: new	133
\@Gls@: added \@glsxtr@record	57		\@print@noidx@glossary: added redefinition	117
\@GLspl@: added \@glsxtr@record ...	57		\glsxtr@addloc@listfield: added group key	12
\@gls@: added \@glsxtr@record	57		added location key	12
\@gls@@link@: added \@glsxtr@record	58		\glsxtr@fields: new	126
\@gls@field@link: added \@glsxtr@record	57		\glsxtr@linkprefix: new	126
\@gls@saveentrycounter: new	26		\glsxtr@org@new@ignored@glossary: new	36
\@glsdisp: added \@glsxtr@record ..	58		\glsxtr@s@new@ignored@glossary: new	36
\@glspl@: added \@glsxtr@record ...	57		\glsxtr@shortcutsval: new	126
\@glsxtr@dorecord: new	10		\glsxtr@texencoding: new	126
\@glsxtr@err@undefaction: new	6		\glsxtr@writefields: new	126
\@glsxtr@record: new	7		\GlsXtrLoadResources: new	126
\@glsxtr@warn@onexists@do: new ...	6		\glsxtr@resourcefile: changed extension to .glstex	125
\@glsxtr@warn@undefaction: new	6		\new@ignored@glossary: added starred version	36
\@print@unsrt@glossary: new	130	1.12 (2017-02-03)		
General: added record package option ...	13		\@glsxtr@recordcounter: new	11
\glsadd: added \@glsxtr@record	62		\@gls@preglossaryhook: check for definition	176
\glsdoifexists: now defines \glslabel	39		\@glsxtr@counter@recordhook: new ..	128
\glsxtr@do@wrglossary: new	26		\@glsxtr@display@loc: new	118
\glsxtr@addloc@listfield: new	12		\@glsxtr@docounter@record: new ...	128
\glsxtr@indexonly@saveentrycounter: new	12		\@glsxtr@longnew@glossaryentry: new	35
\glsxtr@record: new	128		\@glsxtr@noop@recordcounter: new ..	11
\glsxtr@resource: new	126		\@glsxtr@op@recordcounter: new ...	11
\glsxtr@saveentrycounter: new	26		\@glsxtr@provide@storagekey: new ..	27
\glsxtr@setup@record: new	11		\@glsxtr@s@longnew@glossaryentry: new	35
\glsxtr@assignfieldfont: added check for existence	62		\@glsxtrentryfmt: new	30
\glsxtr@resourcefile: new	125		\@glsxtr@index@aliased: new	78
\printunsrt@glossaries: new	130		\@glsxtr@set@alias@noindex: new	77
\printunsrt@glossary: new	130		\@new@glossaryentry@posthook: added check for alias key	45
1.09 (2016-12-16)			\@no@glsxtr@index@aliased: new	78
\@glsxtr@gettype: new	114		\@printunsrt@glossary: new	130
\@glsxtr@mixed@assign@sortkey: new	114			
\@print@glossary: redefined to save options	113			
\glsxtr@make@glossaries: new	114			

General: added target key to printgloss		\GlsXtrLoadResources: removed	
family	113	restriction on only one per document	126
\apptoglossarypreamble: new	34	\glxtrlocrangefmt: new	119
\csGlsXtrLetField: new	33	\glxtrpostlongdescription: new ..	36
\eGlsXtrSetField: new	33	\glxtrprovidestoragekey: new	27
\gGlsXtrSetField: new	33	\GlsXtrRecordCounter: new	128
\glsdohyperlink: added check for alias		\glxtrresourcecount: new	125
field	81	\glxtrresourcefile: added catcode	
\glsnoidxdisplayloc: added		change for @	125
redefinition	118	\glxtrsetaliasnoindex: new	77
\glissettoctitle: added patch	37	\GlsXtrSetField: new	32
\glxtr@counterrecord: new	128	\glxtrsetfieldifexists: new	32
\glxtr@langtag: new	126	\glxtrunsrtdo: new	133
\glxtr@newabbreviation: new	182	\Glsxtrusefield: new	32
\glxtr@org@newignoredglossary:		\glxtrusefield: new	32
Added check for existence	36	short-postlong-user: new	274
\glxtr@pluralsuffixes: new	126	short-postlong-user-desc: new ...	275
\glxtr@provideignoredglossary:		\longnewglossaryentry: added starred	
new	37	version	35
\glxtr@s@newignoredglossary:		long-postshort-user: new	271
Added check for existence	36	long-postshort-user-desc: new ...	273
\glxtr@s@provideignoredglossary:		postdot: new	16
new	38	\pretoglossarypreamble: new	34
\glxtrabbrvpluralsuffix: new ...	187	\print@noop@unsrtglossaryunit:	
\glxtralias: new	44	new	133
\glxtrcopytoglossary: new	38	\print@op@unsrtglossaryunit: new	132
\glxtrdeffield: new	32	\printunsrtglossary: added starred	
\glxtrdisplayendloc: new	119	form	130
\glxtrdisplayendlochook: new ...	119	\printunsrtglossaryhandler: new .	132
\glxtrdisplaysingleloc: new	118	\printunsrtglossaryunit: new	11
\glxtrdisplaystartloc: new	119	\printunsrtglossaryunitsetup: new	132
\glxtrdeffield: new	32	\provideignoredglossary: new	37
\glxtrentryfmt: new	30	\s@glxtr@provide@storagekey: new	28
\glxtrfielddolistloop: new	31	\s@printunsrtglossary: new	130
\glxtrfieldforlistloop: new	31	\xGlsXtrSetField: new	33
\glxtrfieldifinlist: new	31	1.13 (2017-02-07)	
\glxtrfieldlistadd: new	30	\@glsdisp: removed	
\glxtrfieldlistead: new	30	\@glxtr@org@glsdisp	58
\glxtrfieldlistgadd: new	30	\glxtrsetaliasnoindex: switched to	
\glxtrfieldlistxadd: new	30	\providecommand	77
\glxtrfieldxifinlist: new	31	1.14 (2017-04-18)	
\glxtrfmt: new	28	\@gls@link: added redefinition	60
\GlsXtrFmtDefaultOptions: new	28	\@gls@noidx@getgrouptitle: new ..	116
\GlsXtrFmtField: new	28	\@gls@removespaces: new	119
\glxtrifkeydefined: new	27	\@glxtr@do@automake@err: new ...	127
\glxtrindexaliased: new	78	\@glxtr@org@gloautosee: new	25
\GlsXtrLetField: new	33	\@glxtr@record: added third arg	7
\GlsXtrLetFieldToField: new	33	\@glxtr@recordsee: new	11

General: added \glsadd option		1.16 (2017-06-15)	
theHvalue	62	\@glo@autosee: added redefinition	25
added \glsadd option thevalue	62	\@gls@noidx@getgrouptitle: fixed	
\glsdisablehyper: added redefinition .	82	bug	116
\glsenableentrycount: fixed		\@glsxtr@addunusedxrefs: added	
assignment of \@cGls@	94	check for seealso field	46
\glsenableentryunitcount: fixed		\@glsxtr@checkgroup: use \csuse	
assignment of \@cGls@	102	instead of \csname	134
\glsnavigation: new	117	\@glsxtr@dorecordnodefer: new	10
\glsxtr@org@getgrouptitle: new ..	116	\@print@unsrt@glossary: corrected	
\glsxtr@recordsee: new	7	misspelt command	131
\glsxtr@writefields: added check for		\@printunsrt@glossary@handler:	
automake	127	new	132
\glsxtrdisplayendloc: added check		General: added check for	
for empty format	119	\@gls@setupsort@none	14
\glsxtrgetgrouptitle: new	116	\@gls@checkseeallowed: added	
\glsxtrinitwrgloss: new	59	redefinition	25
\glsxtrlocationhyperlink: new ...	120	\@glsxtr@writefields: added	
\glsxtrsetgrouptitle: new	117	\providecommand lines	126
\glsxtrsupphypernumber: new	120	\@glsxtrautoindex: new	170
\ifglsxtrwrglossbefore: new	59	\@glsxtrautoindexassignsort: new .	170
1.15 (2017-05-10)		\@glsxtrautoindexentry: new	170
\@glsxtr@dorecord: corrected		\@glsxtrindexseealso: new	42
premature expansion of \@glslocref	10	\@glsxtrseealsolabels: new	45
short-em-long-em: fixed spelling of		\@glsxtrseelist: new	42
\glsabbrvfont	253	\@glsxtruseseealso: new	41
short-long: fixed spelling of		\@glsxtruseseealsoformat: new	42
\glsabbrvfont	205	\@seealso: new	42
short-long-user: fixed spelling of		autoseeindex: new	15
\glsabbrvfont	277	1.17 (2017-08-09)	
short-postlong-user: fixed spelling of		\@glsxtr@mark@wordseps: new	182
\glsabbrvfont	274	\@glsxtr@markwordseps: new	182
short-postlong-user-desc: fixed		\@glsxtr@noidx@displaynumberlist:	
spelling of \glsabbrvfont	276	replace hard-coded ?? with	
long-em-short-em: fixed spelling of		\glsxtrundeftag	115
\glsabbrvfont	250	\@glsxtr@noidx@entrynumberlist:	
long-postshort-user: fixed spelling of		replace hard-coded ?? with	
\glsabbrvfont	271	\glsxtrundeftag	116
long-postshort-user-desc: fixed		\@glsxtr@noidx@numberlistloop:	
spelling of \glsabbrvfont	273	replace hard-coded ?? with	
long-short: fixed spelling of		\glsxtrundeftag	115
\glsabbrvfont	203	\@glsxtrifhyphenstart: new	279
long-short-user: fixed spelling of		General: removed some inconsistencies	
\glsabbrvfont	270	in the abbreviation styles	203
footnote: fixed spelling of		\glsabbrvhyphenfont: new	279
\glsabbrvfont	208	\glsabbrvonlyfont: new	292
postfootnote: fixed spelling of		\glsabbrvscfont: new	219
\glsabbrvfont	209	\glsabbrvsmfont: new	234

\glsabbrvuserfont: initialised to default font	270	short-long-user-desc: corrected first forms	278
\glsfirstabbrvhyphenfont: new ...	279	short-nolong-desc-noreg: new ...	215
\glsfirstabbrvonlyfont: new	293	short-nolong-noreg: new	213
\glsfirstabbrvscfont: new	219	long-em-noshort-em-desc-noreg: new	265
\glsfirstabbrvsmfont: new	234	long-em-noshort-em-noreg: new ...	262
\glsfirstlonghyphenfont: new ...	280	long-hyphen-noshort-desc-noreg: new	282
\glsfirstlongonlyfont: new	293	long-hyphen-postshort-hyphen: new	285
\glslonghyphenfont: new	279	long-hyphen-postshort-hyphen-desc: new	286
\glslongonlyfont: new	293	long-hyphen-short-hyphen: new ...	280
\glslonguserfont: initialised to default font	270	long-hyphen-short-hyphen-desc: new	281
\glsxtr@newabbreviation: added \glsxtrorgshort and \glsxtrorglong	182	long-noshort-desc-noreg: new ...	218
\GlsXtrDefineAcShortcuts: new	18	long-noshort-noreg: new	219
\glsxtrgenabbrvfmt: added check for \ifglsxtrininsertinside	197	long-only-short-only: new	293
\glsxtrhyphensuffix: new	280	long-only-short-only-desc: new ..	295
\glsxtrifhyphenstart: new	279	long-short-user-desc: corrected first forms	276
\glsxtrlonghyphen: new	284	1.18 (2017-08-10)	
\glsxtrlonghyphennoshort: new ...	281	stylemods: changed default value to "default"	21
\glsxtrlonghyphenshort: new	279	1.19 (2017-09-09)	
\glsxtrlongshortdescname: new ...	204	\@glsxtr@defaultnumberformat: new .	7
\glsxtronlydescname: new	294	\@glsxtr@dorecord: Use \@glsrecordlocref instead of \@glslocref	10
\glsxtronlydescsort: new	294	\@glsxtr@dorecordnodefer: Use \theglsentrycounter for the location rather than \@glslocref ..	10
\glsxtronlysuffix: new	293	\@glsxtr@record@setting: new	12
\glsxtrparen: new	185	\@glsxtr@record@setting@alsoindex: new	13
\glsxtrposthyphenlong: new	290	\@glsxtrifhasfield: new	32
\glsxtrposthyphenshort: new	284	General: added \glslink option theHvalue	59
\glsxtrposthyphensubsequent: new	285	added \glslink option thevalue ...	59
\glsxtrshortdescname: new	213	\glsxtr@writefields: removed double-quotes around \jobname ..	127
\glsxtrshorthyphen: new	289	\glsxtrdoautoindexname: changed format test	169
\glsxtrshorthyphenlong: new	287	\glsxtrhyperlink: new	81
\glsxtrshortlongdescname: new ...	206	\glsxtrifhasfield: new	31
\glsxtrshortlongdescsort: new ...	206	\GlsXtrSetDefaultNumberFormat: new	7
\Glsxtrsubsequentfmt: new	199	\s@glsxtrifhasfield: new	32
\glsxtrsubsequentfmt: new	199		
\Glsxtrsubsequentplfmt: new	199		
\glsxtrsubsequentplfmt: new	199		
\glsxtrword: new	182		
\glsxtrwordsep: new	182		
short-hyphen-long-hyphen: new ...	288		
short-hyphen-long-hyphen-desc: new	289		
short-hyphen-postlong-hyphen: new	290		
short-hyphen-postlong-hyphen-desc: new	292		

1.20 (2017-09-11)		
\@glstrhypernameprefix: new	113	
\glsdohypertarget: added redefinition	114	
\printunstrtglossaryunitsetup:		
switched from redefining		
\glolinkprefix to		
\@glstrhypernameprefix	133	
1.21 (2017-11-03)		
\@glstr@record: added check for		
default options	9	
\@glstrwrglossmark: new	23	
\glslink: changed \let to \def	82	
\@glstr@checkgroup: new	133	
\@glstr@defpostpunc: new	16	
\@glstr@do@record@wrglossary:		
new	8	
\@glstr@dosee@alsoindex@glossary:		
new	25	
\@glstr@doseeglossary: new	24	
\@glstr@noidx@do: removed code		
dealing with the group	135	
\@glstr@record@setting@off: new	13	
\@glstr@record@setting@only: new	13	
\@glstr@rglstrigger@record: new	138	
\@glstrglossentry: new	128	
\@glstrnewgl: new	135	
\@glstrsetaliasnoindex: changed to		
use \glstrifhasfield instead of		
\ifglshasfield	77	
\@glstrwrglossmark: new	23	
\@rGLS: new	141	
\@rGLS@: new	141	
\@rGLSpl: new	141	
\@rGLSpl@: new	142	
\@rGLs: new	140	
\@rGLs@: new	140	
\@rGLspl: new	141	
\@rGLspl@: new	141	
\@rgls: new	139	
\@rgls@: new	139	
\@rglspl: new	140	
\@rglspl@: new	140	
General: adjusted mcolalttree	383	
ac	20	
modified index to remove hard coded		
\space	363	
modified list to remove hard coded		
\space	353	
		moved conditional outside of
		\glsgroupskip
		new
		redefined altlistgroup to discourage
		breaks after group headings
		redefined altlisthypergroup to
		discourage breaks after group
		headings
		redefined alttreegroup to discourage
		breaks after group headings
		redefined alttreehypergroup to
		discourage breaks after group
		headings
		redefined indexgroup to discourage
		breaks after group headings
		redefined indexhypergroup to
		discourage breaks after group
		headings
		redefined listgroup to discourage
		breaks after group headings
		redefined listhypergroup to
		discourage breaks after group
		headings
		redefined mcolalttreegroup to
		discourage breaks after group
		headings
		redefined mcolalttreehypergroup to
		discourage breaks after group
		headings
		redefined mcolalttreespannav to
		discourage breaks after group
		headings
		redefined mcolindexgroup to
		discourage breaks after group
		headings
		redefined mcolindexhypergroup to
		discourage breaks after group
		headings
		redefined mcolindexspannav to
		discourage breaks after group
		headings
		redefined mcoltreegroup to
		discourage breaks after group
		headings
		redefined mcoltreehypergroup to
		discourage breaks after group
		headings
		redefined mcoltreenamegroup to
		discourage breaks after group

headings	382	\glxtrbookindexbetween: new	387
redefined		\glxtrbookindexbookmark: new ...	387
mcoltreenonamehypergroup to		\glxtrbookindexcols: new	386
discourage breaks after group		\glxtrbookindexcolspread: new ..	387
headings	382	\glxtrbookindexfirstmark: new ..	391
redefined mcoltreenonamespannav to		\glxtrbookindexfirstmarkfmt: new	391
discourage breaks after group		\glxtrbookindexformatheader: new	387
headings	383	\glxtrbookindexgroupskip: new ..	387
redefined mcoltreepannav to		\glxtrbookindexlastmark: new ...	391
discourage breaks after group		\glxtrbookindexlastmarkfmt: new	391
headings	381	\glxtrbookindexmarkentry: new ..	391
redefined treegroup to discourage		\glxtrbookindexname: new	386
breaks after group headings	365	\glxtrbookindexparentchildsep:	
redefined treehypergroup to		new	386
discourage breaks after group		\glxtrbookindexparentschildsep:	
headings	366	new	386
redefined treenonamegroup to		\glxtrbookindexprelocation: new	386
discourage breaks after group		\glxtrbookindexsubatendgroup:	
headings	367	new	387
redefined treenonamehypergroup to		\glxtrbookindexsubbetween: new ..	387
discourage breaks after group		\glxtrbookindexsubname: new	386
headings	367	\glxtrbookindexsubprelocation:	
debug: new	23	new	386
\gglsetwidest: new	368	\glxtrbookindexsubsubatendgroup:	
\gl:disablehyper: added check for		new	387
existence	82	\glxtrbookindexsubsubbetween:	
changed to use \def rather than \let .	82	new	387
\gl:enablehyper: changed to use \def		\glxtrbookindexthepage: new	390
rather than \let	82	\glxtrdetoklocation: new	138
\Glsfmtname: new	309	\glxtrenablerecordcount: new ...	138
\gl:fmtname: new	309	\glxtrglossentry: new	128
\glshex: new	316	\glxtrgroupfield: new	133
\glslstchildpostlocation: new ..	353	\GlsXtrheadname: new	301
\glslstchildprelocation: new ...	353	\glxtrheadname: new	300
\glslstprelocation: new	353	\GlsXtrIfFieldEqStr: new	33
\gl:navhyperlink: patched	80	\glxtriflabelinlist: new	132
\gl:seeitemformat: new	41	\glxtrifrecordtrigger: new	138
\gl:showtarget: new	24	\glxtrindexseealso: added check	
\gl:treechildprelocation: new ...	363	that the entry exists	42
\gl:treeprelocation: new	363	\glxtrinithyperoutside: new	59
\gl:triggerrecordformat: new	139	\GlsXtrLocationRecordCount: new ..	137
\gl:useabbrvfont: new	196	\glxtrnewgl: new	135, 136
\gl:uselongfont: new	197	\glxtrnewGLSlike: new	136
\glxtr@do@alsoindex@wrglossary:		\glxtrnewglsl: new	136
new	8	\glxtrnewrgls: new	137
\glxtr@org@@do@wrglossary: new ..	26	\glxtrnewrGLSlike: new	137
\glxtr@org@dohyperlink: new	80	\glxtrnewrglslike: new	137
\glxtr@setbookindexmark: new ...	391	\glxtrprelocation: new	352, 386
\glxtrbookindexatendgroup: new ..	387	\GlsXtrRecordCount: new	137

\glxtrrecordtriggervalue: new ..	138	\glseeitemformat: switched check	
\glxtrresourcefile: now disables		from regular to short	41
record key	125	\glxtr@setaccessdisplay: new ...	167
\glxtrresourceinit: new	125	\glxtr@writefields: provide	
\GlsXtrSetRecordCountAttribute:		\glxtr@record in aux file	126
new	138	\glxtractivatenopost: new	112
\glxtrtitlename: new	301	\glxtrbookindexprelocation:	
\glxtrtitleorpdforheading: new ..	297	removed check for no post dot	386
\GlsXtrTotalRecordCount: new	137	\glxtrglossentryother: new	129
\glxtrwrglossmark: new	23	\glxtrnopostpunc: new	112
short-em: new	256	1.23 (2017-11-12)	
short-sc: corrected first letter		\@@glxtrfmt: added check for indexing	29
uppercasing	224	added grouping	29
short-sm: corrected first letter		new	29
uppercasing	238	\@glxtr@nopostpunc@postdesc: new	113
\ifglxtr@hyperoutside: new	59	\@glxtr@restore@postpunc: new ..	113
all: new	351	\@glxtrentryfmt: fixed missing label	
nolong-short: new	215	argument	30
nolong-short-em: new	258	\@glxtrfmt: new	29
nolong-short-noreg: new	216	\eglupdatewidest: new	369
nolong-short-sc: new	226	\gglupdatewidest: new	369
nolong-short-sm: new	240	\glupdatewidest: new	369
nopostdot: new	16	\GlsXtrDefineAbbreviationShortcuts:	
postpunc: new	16	changed \newabbr definition to use	
\printunsrtglossaryentryprocesshook:		\providecommand	18
new	132	\GlsXtrDefineAcShortcuts: changed	
\printunsrtglossarypredoglossary:		\newabbr definition to use	
new	132	\providecommand	19
\rGLS: new	141	\glxtrfmtdisplay: new	29
\rGLs: new	140	\glxtrifcustomdiscardperiod: new	177
\rgls: new	139	\GlsXtrIfFieldUndef: new	32
\rGLSformat: new	142	\glxtrrestorepostpunc: new	113
\rGLsformat: new	142	\s@glxtrfmt: new	29
\rglsformat: new	142	\s@glxtrfmt: new	29
\rGLSpl: new	141	\xglupdatewidest: new	369
\rGLspl: new	141	1.24 (2017-11-14)	
\rglspl: new	140	\glsadd: added \@gls@setsort	62
\rGLSplformat: new	142	\glxtrforcsvfield: new	31
\rGLsplformat: new	142	\glxtrlocalsetgrouptitle: new ..	117
\rglsplformat: new	142	1.25 (2017-11-14)	
\s@glxtrifhasfield: switched from		\glxtrbookindexmulticolseenv: new	387
\ifdef to \ifundef	32	1.25 (2017-11-24)	
1.22 (2017-11-08)		\glsextrapostnamehook: new	166
\@glxtr@nopostpunc: new	112	\glxtrfootnotename: new	207
\@glxtr@orgprintglossary: changed		\glxtrlongnoshortdescname: new ..	216
explicit \let for \nopostdesc to		\glxtrlongnoshortname: new	218
\glxtractivatenopost	112	\glxtrlongshortname: new	203
\@glxtrglossentryother: new	129	\glxtrlongshortuserdescname: new	273
\glossentrynameother: new	167	\glxtronlyname: new	293

\glxtrpostlinkAddDescOnFirstUse: changed to use \glxtrparen	177	\glxtrGeneralLatinVIrules: new ..	329
\glxtrpostlinkAddSymbolOnFirstUse: changed to use \glxtrparen	178	\glxtrGeneralLatinVrules: new ..	328
\glxtrshortlongname: new	205	\glxtrgeneralpuncIIrules: new ..	325
\glxtrshortlonguserdescname: new	275	\glxtrgeneralpuncIrules: new ...	324
\glxtrshortnolongname: new	211	\glxtrgeneralpuncrules: new	323
1.26 (2018-01-05)		\glxtrhyphenrules: new	323
\@glxtr@do@inc@linkcount: new ..	143	\glxtrLatinA: new	331
\glslinkpresetkeys: new	60	\glxtrLatinAA: new	333
\glxtr@inc@linkcount: new	59	\glxtrLatinAELigature: new	333
\GlsXtrEnableLinkCounting: new ..	144	\glxtrLatinE: new	331
\GlsXtrIfLinkCounterDef: new	144	\glxtrLatinEszettSs: new	332
\glxtrinclinkcounter: new	143	\glxtrLatinEszettSz: new	332
\GlsXtrLinkCounterName: new	144	\glxtrLatinEth: new	333
\GlsXtrLinkCounterValue: new	143	\glxtrLatinH: new	331
\GlsXtrTheLinkCounter: new	144	\glxtrLatinI: new	331
1.27 (2018-02-26)		\glxtrLatinInsularG: new	333
\@glxtrdialecthook: new	26	\glxtrLatinK: new	331
General: added		\glxtrLatinL: new	331
glossaries-extra-bib2gls.sty	315	\glxtrLatinLslash: new	333
\Alpha: new	317	\glxtrLatinM: new	331
\Beta: new	317	\glxtrLatinN: new	332
\Chi: new	318	\glxtrLatinO: new	332
\Digamma: new	318	\glxtrLatinOELigature: new	333
\Epsilon: new	317	\glxtrLatinOslash: new	333
\Eta: new	317	\glxtrLatinP: new	332
\glxtr@loaddialect: new	315	\glxtrLatinS: new	332
\glxtrBasicDigitrules: new	348	\glxtrLatinSchwa: new	332
\glxtrcombiningdiacriticIIIrules: new	322	\glxtrLatinT: new	332
\glxtrcombiningdiacriticIIrules: new	321	\glxtrLatinThorn: new	333
\glxtrcombiningdiacriticIrules: new	321	\glxtrLatinWynn: new	333
\glxtrcombiningdiacriticIVrules: new	323	\glxtrLatinX: new	332
\glxtrcombiningdiacriticrules: new	321	\glxtrMathGreekIIrules: new	340
\glxtrcontrolrules: new	320	\glxtrMathGreekIrules: new	338
\glxtrcurrencyrules: new	324	\glxtrMathItalicAlpha: new	344
\glxtrdigitrules: new	347	\glxtrMathItalicBeta: new	344
\glxtrfractionrules: new	348	\glxtrMathItalicChi: new	347
\glxtrGeneralLatinIIIrules: new	326	\glxtrMathItalicDelta: new	344
\glxtrGeneralLatinIIrules: new ..	326	\glxtrMathItalicEpsilon: new ...	344
\glxtrGeneralLatinIrules: new ..	325	\glxtrMathItalicEta: new	345
\glxtrGeneralLatinIVrules: new ..	327	\glxtrMathItalicGamma: new	344
\glxtrGeneralLatinVIIrules: new	330	\glxtrMathItalicGreekIIrules: new	335
\glxtrGeneralLatinVIIrules: new	329	\glxtrMathItalicGreekIrules: new	335
		\glxtrMathItalicIota: new	345
		\glxtrMathItalicKappa: new	345
		\glxtrMathItalicLambda: new	345
		\glxtrMathItalicLowerGreekIIrules: new	338

<code>\glxtrMathItalicLowerGreekIrules:</code>	<code>\glxtrUpTau:</code>	new	343
new	<code>\glxtrUpTheta:</code>	new	342
<code>\glxtrMathItalicMu:</code>	<code>\glxtrUpUpsilon:</code>	new	343
new	<code>\glxtrUpXi:</code>	new	342
<code>\glxtrMathItalicNabla:</code>	<code>\glxtrUpZeta:</code>	new	341
new	<code>\Iota:</code>	new	318
<code>\glxtrMathItalicNu:</code>	<code>\Kappa:</code>	new	318
new	<code>\Mu:</code>	new	318
<code>\glxtrMathItalicOmega:</code>	<code>\Nu:</code>	new	318
new	<code>\Omicron:</code>	new	318
<code>\glxtrMathItalicOmicron:</code>	<code>\omicron:</code>	new	318
new	<code>\Rho:</code>	new	318
<code>\glxtrMathItalicPartial:</code>	<code>\Tau:</code>	new	318
new	<code>\Upalpha:</code>	new	318
<code>\glxtrMathItalicPhi:</code>	<code>\Upbeta:</code>	new	318
new	<code>\Upchi:</code>	new	319
<code>\glxtrMathItalicPi:</code>	<code>\Upsilon:</code>	new	318
new	<code>\Upeta:</code>	new	319
<code>\glxtrMathItalicPsi:</code>	<code>\Upiota:</code>	new	319
new	<code>\Upkappa:</code>	new	319
<code>\glxtrMathItalicRho:</code>	<code>\Upmu:</code>	new	319
new	<code>\Upnu:</code>	new	319
<code>\glxtrMathItalicSigma:</code>	<code>\Upomicron:</code>	new	319
new	<code>\upomicron:</code>	new	319
<code>\glxtrMathItalicTau:</code>	<code>\Uprho:</code>	new	319
new	<code>\Uptau:</code>	new	319
<code>\glxtrMathItalicTheta:</code>	<code>\Upzeta:</code>	new	318
new	<code>\Zeta:</code>	new	317
<code>\glxtrMathItalicUpperGreekIrules:</code>			
new			
<code>\glxtrMathItalicUpperGreekIrules:</code>			
new			
<code>\glxtrMathItalicUpsilon:</code>			
new			
<code>\glxtrMathItalicXi:</code>			
new			
<code>\glxtrMathItalicZeta:</code>			
new			
<code>\glxtrMathUpGreekIrules:</code>			
new			
<code>\glxtrMathUpGreekIrules:</code>			
new			
<code>\glxtrnonprintablerules:</code>			
new			
<code>\glxtrprovidecommand:</code>			
new			
<code>\glxtrspacerules:</code>			
new			
<code>\glxtrSubScriptDigitrules:</code>			
new			
<code>\glxtrSuperScriptDigitrules:</code>			
new			
<code>\glxtrUpAlpha:</code>			
new			
<code>\glxtrUpBeta:</code>			
new			
<code>\glxtrUpChi:</code>			
new			
<code>\glxtrUpDelta:</code>			
new			
<code>\glxtrUpDigamma:</code>			
new			
<code>\glxtrUpEpsilon:</code>			
new			
<code>\glxtrUpEta:</code>			
new			
<code>\glxtrUpGamma:</code>			
new			
<code>\glxtrUpIota:</code>			
new			
<code>\glxtrUpKappa:</code>			
new			
<code>\glxtrUpLambda:</code>			
new			
<code>\glxtrUpMu:</code>			
new			
<code>\glxtrUpNu:</code>			
new			
<code>\glxtrUpOmega:</code>			
new			
<code>\glxtrUpOmicron:</code>			
new			
<code>\glxtrUpPhi:</code>			
new			
<code>\glxtrUpPi:</code>			
new			
<code>\glxtrUpPsi:</code>			
new			
<code>\glxtrUpRho:</code>			
new			
<code>\glxtrUpSigma:</code>			
new			

1.28 (2018-03-06)

<code>\@glxtr@docdefval:</code>	changed from	
count register to macro		14
<code>\@glxtr@dialecthook:</code>	save and restore	
<code>\TrackLangRequireDialectPrefix</code>		
		349
<code>\glxtr@redefeffield:</code>	changed <code>\csedef</code> to	
<code>\protected@csedef</code>		32
<code>\glxtr@localsetgrouptitle:</code>	changed	
<code>\csedef \protected@csedef</code>		117
<code>\glxtr@rsetgrouptitle:</code>	changed	
<code>\csxdef \protected@csxdef</code>		117

1.29 (2018-04-09)

<code>\@glxtr@removespaces:</code>	added expansion	120
<code>\@glxtr@dorecord:</code>	check for page	
counter		10
<code>\@glxtr@wrglossary@locationhyperlink:</code>		
new		22
<code>\glxtr@inc@wrglossaryctr:</code>	new	22
<code>\glxtr@wrglossarylocation:</code>	new	316
<code>\GlsXtrBibTeXEntryAliases:</code>	new	316

\glstrfieldforlistloop:corrected	new	22
argument order in \forlistcsloop	\GlsXtrProvideBibTeXFields:new	317
\GlsXtrIndexCounterLink:new	indexcounter:new	23
\GlsXtrInternalLocationHyperlink:	\setentrycounter:new	119
		316

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	
\.	16, 177, 363
\@	47, 125
\@cGLS@	94, 102
\@cGLSpl@	94, 102
\@cGls@	94, 102
\@cGlspl@	94, 102
\@cgls@	94, 102
\@cglspl@	94, 102
\@do@wrglossary	8–10, 109
\@do@wrglossary	11, 13, 14, 26, 62, 78
\@glo@assign@sortkey	114
\@glo@list	6
\@glo@type	130
\@glossarysec	387
\@gls@expand@field	28
\@glslocalreset	92
\@glslocalunset	91
\@glsreset	92
\@glsunset	91
\@glsxtr@autoindex@escspch	171–173
\@glsxtr@checkspch	170, 171, 173
\@glsxtr@disabledflycommand	51, 52
\@glsxtr@org@postdescription	112, 113
\@glsxtr@record	13, 14
\@glsxtr@recordcounter	13, 14, 128
\@glsxtrfmt	29
\@glsxtrp	86, 87
\@glsxtrpostloctag	54
\@glsxtrpreloctag	54
\@glsxtrwrglossmark	8, 9, 11, 25, 26, 43, 47, 108
\@newglossaryentry@defcounters	93
\@newglossaryentry@defunitcounters	101
\@par	368
\@ACRlong	83
\@ACRlongpl	83
\@ACRshort	83
\@ACRshortpl	83
\@Acrlong	83
\@Acrlongpl	83
\@Acrshort	83
\@Acrshortpl	83
\@GLS@	83, 96, 97, 141
\@GLSdesc@	67
\@GLSpl@	83, 97, 98, 142
\@GLSplural@	84
\@GLSsymbol@	68
\@GLStext@	83
\@GLSxtr@full	188
\@GLSxtr@fullpl	189
\@GLSxtr@p@acrlong@	83
\@GLSxtr@p@acrlongpl@	83
\@GLSxtr@p@acrshort@	83
\@GLSxtr@p@acrshortpl@	83
\@GLSxtr@p@long@	83
\@GLSxtr@p@longpl@	83
\@GLSxtr@p@plural@	83
\@GLSxtr@p@short@	83
\@GLSxtr@p@shortpl@	83
\@GLSxtr@p@text@	83
\@GLSxtrlong	83, 193
\@GLSxtrlongpl	83, 196
\@GLSxtrp	90, 91
\@GLSxtrshort	83, 191
\@GLSxtrshortpl	83, 194
\@Gls@	83, 96, 97, 140
\@Gls@acentryname	105
\@Gls@entry@field	74, 89, 90, 168
\@Gls@entryname	105
\@GlsXtrEnableOnTheFly	48, 49
\@Glspl@	83, 96, 97, 141
\@Glsplural@	84
\@Glstext@	83

\@Glsxtr	50, 51	\@end@glxstrifhyphenstart	279
\@Glsxtr@full	187	\@endfortrue	31, 167, 200
\@Glsxtr@fullpl	189	\@firstofone	63, 131, 161, 162, 169, 175
\@Glsxtr@p@acrlong@	83	\@firstofthree	58, 62, 70–73, 79, 80, 187, 189, 190, 192, 193, 195
\@Glsxtr@p@acrlongpl@	83	\@firstoftwo 64, 65, 67, 68, 71–74, 76, 79, 107, 167, 179, 180, 187, 189, 193–196, 297, 298
\@Glsxtr@p@acrshort@	83	\@for	6, 21, 31, 46, 93, 104, 108, 111, 117, 131, 138, 144, 160, 167, 174
\@Glsxtr@p@acrshortpl@	83	\@glo@alias	43, 44
\@Glsxtr@p@long@	83	\@glo@assign@sortkey	110
\@Glsxtr@p@longpl@	83	\@glo@autosee	25
\@Glsxtr@p@plural@	83	\@glo@autoseehook	44
\@Glsxtr@p@short@	83	\@glo@category	98
\@Glsxtr@p@shortpl@	83	\@glo@check@sortallowed	111
\@Glsxtr@p@text@	83	\@glo@counterprefix	10, 11, 119, 120
\@Glsxtrlong	83, 192	\@glo@countunit	98, 99
\@Glsxtrlongpl	83, 195	\@glo@default@sorttype	110
\@Glsxtrp	89, 90	\@glo@desc	35
\@Glsxtrpl	51	\@glo@descplural	35
\@Glsxtrshort	83, 190	\@glo@group	12
\@Glsxtrshortpl	83, 194	\@glo@label	12, 28, 40, 43–45, 74, 82, 370–377
\@acrlong	83	\@glo@location	12
\@acrlongpl	83	\@glo@loclist	12
\@acrshort	83	\@glo@name	169, 170
\@acrshortpl	83	\@glo@no@assign@sortkey	114
\@addtoreset	143	\@glo@parent	372, 373
\@afterheading 354, 355, 364–367, 380–383, 390	\@glo@see	40–42, 44–46
\@alt@glshyp@opt	79	\@glo@seealso	43, 44
\@auxout	10, 11, 47, 55, 94, 103, 108, 109, 121, 125–128, 391	\@glo@sort	170
\@bibgl@restoreat	125	\@glo@sorttype	110, 117
\@cGLS	97	\@glo@text	58
\@cGLS@	94, 97, 102	\@glo@thislettergrp	134
\@cGLSpl	97	\@glo@thisvalue	270
\@cGLSpl@	94, 97, 102	\@glo@tmp	28, 42, 74
\@cGls@	94, 102	\@glo@type	45, 80, 105, 108, 111, 112, 114, 117, 118, 120, 121, 124, 125, 131
\@cGlspl@	94, 102	\@glo@types	158, 159, 370–376
\@cgls@	94, 102	\@glossary@default@style	52, 111, 385
\@cglspl@	94, 102	\@glossarystyle	111, 112
\@disable@onlypremakeg	108	\@gls@	83, 95, 97, 140
\@do@auxoutstuff	121	\@gls@@link	58
\@do@gl@getcounterprefix	10, 11	\@gls@ReturnAfterFi	120
\@do@gl@see	44, 45	\@gls@actualchar	170
\@do@newglossaryentry	105, 184	\@gls@adjustmode	62
\@do@seeglossary	13, 14, 24, 47, 108	\@gls@alt@hyp@opt	80
\@do@wrglossary	61, 139	\@gls@alt@hyp@opt@char	79, 80
\@empty	62, 70–74, 113, 119, 170, 171, 187–196	\@gls@alt@hyp@opt@keys	80
\@end@glxtr@addunused	46		
\@end@glxtr@gettype	110, 114		
\@end@glxtr@usese	41		

<code>\@gls@automake</code>	111	<code>\@gls@noidx@sanitizesort</code>	110
<code>\@gls@between</code>	117	<code>\@gls@noidxloclist@finalsep</code> ...	114, 115
<code>\@gls@checkedmkidx</code>	170, 171, 173	<code>\@gls@noidxloclist@prev</code>	114, 115
<code>\@gls@checkmkidxchars</code>	43, 170	<code>\@gls@noidxloclist@sep</code>	114
<code>\@gls@codepage</code>	121	<code>\@gls@noref@warn</code>	109, 118
<code>\@gls@counter</code>	9–11, 60, 62, 78, 139	<code>\@gls@org@glsnoidxdisplayloc</code>	115
<code>\@gls@currentlettergroup</code> ...	118, 131, 134	<code>\@gls@org@glsseeformat</code>	115
<code>\@gls@declareoption</code>	5	<code>\@gls@preglossaryhook</code>	112, 174
<code>\@gls@default@longpl</code>	183, 184	<code>\@gls@prevlevel</code>	378, 379, 383, 384
<code>\@gls@doautomake</code>	111, 127	<code>\@gls@quotechar</code>	170
<code>\@gls@doautomake@err</code>	127	<code>\@gls@reference</code>	47, 108, 109
<code>\@gls@enablesavenonumberlist</code>	46	<code>\@gls@restreat</code>	47
<code>\@gls@encapchar</code>	171	<code>\@gls@saveentrycounter</code> 13, 14, 26, 60, 62, 139	
<code>\@gls@entry@count</code>	94	<code>\@gls@see@noindex</code>	25, 125
<code>\@gls@entry@field</code>		<code>\@gls@setdefault@glslink@opts</code>	
.....	28, 32, 44, 74, 88–91, 93, 129, 130	9, 29, 60, 78
<code>\@gls@entry@unitcount</code>	102, 103	<code>\@gls@setsort</code>	61, 62
<code>\@gls@field@font</code>	63–70	<code>\@gls@setupsort@none</code>	14
<code>\@gls@field@link</code>	63–70, 75, 76	<code>\@gls@short</code>	183, 184
<code>\@gls@getcounterprefix</code>	10, 11	<code>\@gls@shortpl</code>	181, 183, 184
<code>\@gls@getgrouptitle</code>	116, 117, 131	<code>\@gls@sort</code>	134
<code>\@gls@grptitle</code>	80, 117	<code>\@gls@thisval</code>	167
<code>\@gls@hyp@opt</code>		<code>\@gls@tmp</code>	117
.....	75, 76, 80, 97, 136, 139–141, 187–196	<code>\@gls@tmpb</code>	173
<code>\@gls@hyp@opt@cs</code>	79, 80	<code>\@gls@type</code>	108, 109, 111, 200, 370–377
<code>\@gls@ifinlist</code>	132	<code>\@gls@write@entrycounts</code>	94
<code>\@gls@increment@currcount</code>	93	<code>\@gls@write@entryunitcounts</code>	102
<code>\@gls@increment@currunitcount</code>	101	<code>\@gls@write@entryunitcounts@do</code>	103
<code>\@gls@keymap</code>	12, 27, 41, 43, 74, 126, 167	<code>\@gls@xref</code>	11, 43
<code>\@gls@label</code> ..	8–11, 47, 79, 108, 109, 128, 200	<code>\@gls@abbrv@current@abbreviation</code> 182, 196	
<code>\@gls@levelchar</code>	170	<code>\@gls@acronymlists</code>	105
<code>\@gls@link</code>	29, 57, 58, 70–74, 187–196	<code>\@gls@doifexistsorwarn</code>	15, 163–167
<code>\@gls@link@checkfirsthyper</code>	58, 107	<code>\@gls@entry</code>	94, 103
<code>\@gls@link@label</code>	60, 139	<code>\@gls@link</code>	61, 80, 82
<code>\@gls@link@nocheckfirsthyper</code>		<code>\@gls@nextpages</code>	112
.....	57, 70–74, 187–196	<code>\@gls@nonextpages</code>	112
<code>\@gls@link@opts</code>	60	<code>\@gls@numberformat</code>	
<code>\@gls@list</code>	117	9–11, 60, 62, 78, 139, 166, 169
<code>\@gls@local@increment@currcount</code>	93	<code>\@gls@order</code>	108
<code>\@gls@local@increment@currunitcount</code> 102		<code>\@gls@pl@</code>	83, 96, 97, 140
<code>\@gls@location</code>	134, 135	<code>\@gls@plural@</code>	83
<code>\@gls@loclist</code>	114, 115, 134, 135	<code>\@gls@punc@token</code>	179
<code>\@gls@long</code>	183	<code>\@gls@recordlocref</code>	10
<code>\@gls@longpl</code>	181, 183, 184	<code>\@gls@showtarget</code>	81
<code>\@gls@map</code>	167	<code>\@gls@style@altlist</code>	353
<code>\@gls@nohyperlist</code>	36, 38	<code>\@gls@style@altlistgroup</code>	354
<code>\@gls@noidx@do</code>	118	<code>\@gls@style@altlisthypergroup</code>	355
<code>\@gls@noidx@getgrouptitle</code>	131	<code>\@gls@style@alttree</code>	367
<code>\@gls@noidx@nosanitizesort</code>	110	<code>\@gls@style@alttreegroup</code>	379

<code>\@glsstyle@alttreehypergroup</code>	379	<code>\@glsxtr@autoindex@level</code>	170–172
<code>\@glsstyle@index</code>	363	<code>\@glsxtr@autoindex@setname</code>	169
<code>\@glsstyle@indexgroup</code>	364	<code>\@glsxtr@autoindexcrossrefs</code>	13, 15, 40, 44
<code>\@glsstyle@indexhypergroup</code>	364	<code>\@glsxtr@autoseeindexfalse</code>	13
<code>\@glsstyle@inline</code>	363	<code>\@glsxtr@autoseeindextrue</code>	15
<code>\@glsstyle@list</code>	353	<code>\@glsxtr@bookindex@atendgroup</code> ..	388–390
<code>\@glsstyle@listdotted</code>	352	<code>\@glsxtr@bookindex@atsubendgroup</code> ..	389
<code>\@glsstyle@listgroup</code>	354	<code>\@glsxtr@bookindex@atsubsubendgroup</code> ..	389
<code>\@glsstyle@listhypergroup</code>	354	<code>\@glsxtr@bookindex@between</code>	388, 390
<code>\@glsstyle@mcolalttree</code>	383	<code>\@glsxtr@bookindex@sep</code>	388, 389
<code>\@glsstyle@mcolalttreegroup</code>	383	<code>\@glsxtr@bookindex@subatendgroup</code> ..	
<code>\@glsstyle@mcolalttreehypergroup</code> ..	384	388–390
<code>\@glsstyle@mcolalttreespannav</code>	384	<code>\@glsxtr@bookindex@subbetween</code> ..	388, 389
<code>\@glsstyle@mcolindexgroup</code>	380	<code>\@glsxtr@bookindex@subsep</code>	388, 389
<code>\@glsstyle@mcolindexhypergroup</code>	380	<code>\@glsxtr@bookindex@subsubatendgroup</code> ..	
<code>\@glsstyle@mcolindexspannav</code>	380	388–390
<code>\@glsstyle@mcoltreegroup</code>	381	<code>\@glsxtr@bookindex@subsubbetween</code> ..	
<code>\@glsstyle@mcoltreehypergroup</code>	381	388, 389
<code>\@glsstyle@mcoltreenonamegroup</code>	382	<code>\@glsxtr@bookindexgroupskip</code> ...	388, 390
<code>\@glsstyle@mcoltreenonamehypergroup</code> ..	382	<code>\@glsxtr@cat</code>	93, 104, 138, 174
<code>\@glsstyle@mcoltreenonamespannav</code> ..	383	<code>\@glsxtr@checkgroup</code>	131
<code>\@glsstyle@mcoltreespannav</code>	381	<code>\@glsxtr@counterrecordhook</code>	10, 11
<code>\@glsstyle@tree</code>	365	<code>\@glsxtr@csname</code>	99, 100, 102
<code>\@glsstyle@treegroup</code>	365	<code>\@glsxtr@current@style</code>	52, 385
<code>\@glsstyle@treehypergroup</code>	366	<code>\@glsxtr@currentunitcount</code> ...	99, 100, 102
<code>\@glsstyle@treenoname</code>	366	<code>\@glsxtr@currunitcount</code>	101, 103
<code>\@glsstyle@treenonamegroup</code>	367	<code>\@glsxtr@debugnr</code>	23
<code>\@glsstyle@treenonamehypergroup</code> ...	367	<code>\@glsxtr@debugval</code>	23
<code>\@glstarget</code>	82, 113	<code>\@glsxtr@declareoption</code> ...	5, 15–17, 20, 23
<code>\@glstext@</code>	83	<code>\@glsxtr@defaultnoglossarywarning</code> ..	21
<code>\@glswidestname</code>	370, 377	<code>\@glsxtr@defaultnumberformat</code>	
<code>\@glsxtr</code>	49, 51	7, 9, 60, 62, 78, 166, 169
<code>\@glsxtr@do@wrglossary</code>	109	<code>\@glsxtr@defpostpunc</code>	16, 17, 24
<code>\@glsxtr@abbreviationsdef</code>	17, 25, 26	<code>\@glsxtr@deprecated@abbrstyle</code>	
<code>\@glsxtr@accessdisplay</code>	167, 168	228, 230, 232,
<code>\@glsxtr@activate@initialtagging</code>	233, 243, 244, 246, 248, 260, 264, 267, 269
.....	174, 176	<code>\@glsxtr@disabledflycommand</code>	51
<code>\@glsxtr@addunitcounter</code>	99	<code>\@glsxtr@display@loc</code>	118
<code>\@glsxtr@addunused</code>	46	<code>\@glsxtr@do@windex</code>	79
<code>\@glsxtr@addunusedxrefs</code>	45, 46	<code>\@glsxtr@do@gl:disablehyperinlist</code> ..	77
<code>\@glsxtr@attrval</code>	61, 161–169	<code>\@glsxtr@do@inc@linkcount</code>	144
<code>\@glsxtr@autoindex@at</code>	170, 171	<code>\@glsxtr@do@record@wrglossary</code>	8, 13
<code>\@glsxtr@autoindex@doextra@esc</code>	170	<code>\@glsxtr@do@redef@for@gl@sentries</code>	7
<code>\@glsxtr@autoindex@encap</code>	169, 171	<code>\@glsxtr@do@style</code>	22, 315
<code>\@glsxtr@autoindex@esc</code>	170, 172, 173	<code>\@glsxtr@do@titlecaps@warn</code>	
<code>\@glsxtr@autoindex@escat</code>	170, 171	161–164, 168, 175
<code>\@glsxtr@autoindex@escencap</code>	171	<code>\@glsxtr@doabbreviationsdef</code>	17
<code>\@glsxtr@autoindex@esclevel</code> ...	171, 172	<code>\@glsxtr@doaccsupp</code>	20, 24
<code>\@glsxtr@autoindex@escquote</code> ...	170, 172	<code>\@glsxtr@docdefsetting</code>	15

\@glxtr@docdefval	14, 15, 46, 48	\@glxtr@mixed@assign@sortkey	110
\@glxtr@doccounterrecord	11	\@glxtr@noidx@displaynumberlist ..	109
\@glxtr@doglossary	131	\@glxtr@noidx@do	133
\@glxtr@doiflabelinlist	132	\@glxtr@noidx@entrynumberlist	110
\@glxtr@doloctag	54, 55	\@glxtr@noidx@numberlistloop	110
\@glxtr@dorecord	8, 9	\@glxtr@nomissingglstextnr	21
\@glxtr@dorecordnodefer	8, 10	\@glxtr@nomissingglstextval	21
\@glxtr@dosee@alsoindex@glossary ..	14	\@glxtr@noop@recordcounter	11, 13
\@glxtr@doseeglossary	13, 25	\@glxtr@nopostpunc	112
\@glxtr@dostylewarn	200	\@glxtr@nopostpunc@postdesc ..	112, 113
\@glxtr@enabletagging	174	\@glxtr@notfoundinlist	180
\@glxtr@end@	49	\@glxtr@op@recordcounter	13, 14
\@glxtr@endescspch	170–173	\@glxtr@optlist	51
\@glxtr@entrycount@org@localreset ..	94	\@glxtr@org@starttoc	296
\@glxtr@entrycount@org@localunset ..	93	\@glxtr@org@GLS@	58
\@glxtr@entrycount@org@reset	94	\@glxtr@org@GLSpl@	58
\@glxtr@entrycount@org@unset	93	\@glxtr@org@Gls@	57
\@glxtr@entryunitcount@org@localreset	102	\@glxtr@org@Glspl@	57
\@glxtr@entryunitcount@org@localunset	101, 102	\@glxtr@org@Glsxtrtitlefirst ..	297, 298
\@glxtr@entryunitcount@org@reset ..	102	\@glxtr@org@Glsxtrtitlefirstplural	297, 299
\@glxtr@entryunitcount@org@unset ..	101	\@glxtr@org@Glsxtrtitlefull ..	297, 299
\@glxtr@err@undefaction	7, 13	\@glxtr@org@Glsxtrtitlefullpl ..	297, 299
\@glxtr@field@linkdefs	57	\@glxtr@org@Glsxtrtitlelong ..	297, 299
\@glxtr@format@overridefalse	168	\@glxtr@org@Glsxtrtitlelongpl ..	297, 299
\@glxtr@format@overridefalse	169	\@glxtr@org@Glsxtrtitlename ..	297, 298
\@glxtr@foundinlist	180	\@glxtr@org@Glsxtrtitleplural ..	297, 298
\@glxtr@full	187	\@glxtr@org@Glsxtrtitleshort ..	297, 298
\@glxtr@fullpl	188	\@glxtr@org@Glsxtrtitleshortpl ..	297, 298
\@glxtr@gettype	110	\@glxtr@org@Glsxtrtitletext ..	297, 298
\@glxtr@glossdescfont	161, 162	\@glxtr@org@MakeUppercase	297, 298
\@glxtr@glossnamefont	163–168	\@glxtr@org@checkfirsthyper ...	76, 107
\@glxtr@gobbleto@endescspch	173	\@glxtr@org@delimN	54, 55
\@glxtr@groupheading	131, 134	\@glxtr@org@delimR	54, 55
\@glxtr@idx@displaynumberlist	109	\@glxtr@org@doseeglossary	25, 108
\@glxtr@idx@entrynumberlist	110	\@glxtr@org@gloautoosee	25
\@glxtr@ifcsstart	49	\@glxtr@org@gls@	57
\@glxtr@ifpunctoken	179	\@glxtr@org@glsdohypertarget	114
\@glxtr@ifunitcounter	98	\@glxtr@org@glsignore	54, 55
\@glxtr@insert@dots	181	\@glxtr@org@glspl@	57
\@glxtr@insert@dots@next	181	\@glxtr@org@glsxtrtitlefirst ..	297, 298
\@glxtr@insert@dots	183	\@glxtr@org@glsxtrtitlefirstplural	297, 298
\@glxtr@label	31, 46, 144, 160	\@glxtr@org@glsxtrtitlefull ..	297, 299
\@glxtr@loadstyles	351, 352	\@glxtr@org@glsxtrtitlefullpl ..	297, 299
\@glxtr@longnewglossaryentry	35	\@glxtr@org@glsxtrtitlelong ..	297, 299
\@glxtr@mark@wordseps	182	\@glxtr@org@glsxtrtitlelongpl ..	297, 299
\@glxtr@mark@wordseps@next	182	\@glxtr@org@glsxtrtitlename ..	297, 298
\@glxtr@mark@wordseps	183, 184		

<code>\@glsxtr@org@glsxtrtitleorpdforheading</code>	<code>\@glsxtr@rglstrigger@record</code> ...	140–142
..... 297, 298	<code>\@glsxtr@s@longnewglossaryentry</code>	35
<code>\@glsxtr@org@glsxtrtitleplural</code> 297, 298	<code>\@glsxtr@savepreloctag</code>	54, 55
<code>\@glsxtr@org@glsxtrtitleshort</code> . 297, 298	<code>\@glsxtr@setentrycountunsetattr</code>	92
<code>\@glsxtr@org@glsxtrtitleshortpl</code> 297, 298	<code>\@glsxtr@setentryunitcountunsetattr</code> 104	
<code>\@glsxtr@org@glsxtrtitletext</code> .. 297, 298	<code>\@glsxtr@setupshortcuts</code>	19, 20, 25, 26
<code>\@glsxtr@org@makeglossaries</code>	<code>\@glsxtr@shortcutsnr</code>	19
<code>\@glsxtr@org@markboth</code>	<code>\@glsxtr@shortcutsval</code>	19, 127
<code>\@glsxtr@org@markright</code>	<code>\@glsxtr@swaptwo</code>	180
<code>\@glsxtr@org@newacronymstyle</code>	<code>\@glsxtr@tag</code>	174
<code>\@glsxtr@org@postdescription</code> .. 113, 176	<code>\@glsxtr@taggingcs</code>	174
<code>\@glsxtr@org@see@noindex</code>	<code>\@glsxtr@textformat</code>	61
<code>\@glsxtr@org@setacronymstyle</code>	<code>\@glsxtr@theHvalue</code>	8, 9, 59, 60, 62, 139
<code>\@glsxtr@org@theHvalue</code>	<code>\@glsxtr@thevalue</code>	8, 9, 59, 60, 62, 139
<code>\@glsxtr@org@prefix</code>	<code>\@glsxtr@thisloctag</code>	55
<code>\@glsxtr@org@printglossary</code>	<code>\@glsxtr@titlelabel</code>	116, 117, 133
<code>\@glsxtr@org@warndep</code>	<code>\@glsxtr@tmp</code>	21, 22, 119
<code>\@glsxtr@p@acrlong@</code>	<code>\@glsxtr@type</code>	160
<code>\@glsxtr@p@acrlongpl@</code>	<code>\@glsxtr@unitcountlist</code>	99
<code>\@glsxtr@p@acrshort@</code>	<code>\@glsxtr@unsrt@getgroupptitle</code>	131
<code>\@glsxtr@p@acrshortpl@</code>	<code>\@glsxtr@usesee</code>	41
<code>\@glsxtr@p@long@</code>	<code>\@glsxtr@warn@onexistsordo</code>	7, 13, 14
<code>\@glsxtr@p@longpl@</code>	<code>\@glsxtr@warn@undefaction</code>	7, 13, 14
<code>\@glsxtr@p@plural@</code>	<code>\@glsxtr@wrglossary@locationhyperlink</code>	
<code>\@glsxtr@p@short@</code>	23
<code>\@glsxtr@p@shortpl@</code>	<code>\@glsxtr@wrglossnr</code>	59
<code>\@glsxtr@p@text@</code>	<code>\@glsxtr@wrglossval</code>	59
<code>\@glsxtr@pagetag</code>	<code>\@glsxtr@dialecthook</code>	315
<code>\@glsxtr@pagetag</code>	<code>\@glsxtr@docdeffalse</code>	48
<code>\@glsxtr@prevunitcount</code>	<code>\@glsxtr@entryfmt</code>	30
<code>\@glsxtr@printglossnr</code>	<code>\@glsxtr@fmt</code>	29
<code>\@glsxtr@printglossopts</code>	<code>\@glsxtr@glossentry</code>	128
<code>\@glsxtr@printglossval</code>	<code>\@glsxtr@glossentryother</code>	129
<code>\@glsxtr@printunsrtglossaryskipentry</code>	<code>\@glsxtr@hypernameprefix</code>	113, 114, 133
..... 131, 132	<code>\@glsxtr@ifhasfield</code>	31
<code>\@glsxtr@provide@addstoragekey</code>	<code>\@glsxtr@ifhyphenstart</code>	279
<code>\@glsxtr@provide@storagekey</code>	<code>\@glsxtr@indexaliased</code>	77
<code>\@glsxtr@record</code>	<code>\@glsxtr@indexcrossrefsfalse</code>	15
<code>\@glsxtr@record@setting</code>	<code>\@glsxtr@indexcrossrefstrue</code>	15
..... 8, 10, 13, 42, 47, 48, 107, 108	<code>\@glsxtr@inmark</code>	296
<code>\@glsxtr@record@setting@alsoindex</code> .	<code>\@glsxtr@long</code>	83, 192
..... 8, 10, 42, 108	<code>\@glsxtr@longpl</code>	83, 195
<code>\@glsxtr@record@setting@off</code>	<code>\@glsxtr@newgls</code>	136, 137
<code>\@glsxtr@record@setting@only</code>	<code>\@glsxtr@newgls@inner</code>	135, 136
<code>\@glsxtr@recordsee</code>	<code>\@glsxtr@newgls@innercsname</code>	136
<code>\@glsxtr@redef@forglsentries</code>	<code>\@glsxtr@notinmark</code>	296
<code>\@glsxtr@redef@styles</code>	<code>\@glsxtr@p</code>	88, 89
<code>\@glsxtr@reg@glosslist</code>	<code>\@glsxtr@p@opt</code>	86
<code>\@glsxtr@restore@postpunc</code>	<code>\@glsxtr@pl</code>	50, 51
..... 112, 113		

	243, 245, 247, 248, 250, 252, 254, 255, 257, 259, 260, 262, 264, 266, 268, 271, 272, 274, 277, 280, 282, 285, 288, 291, 293		
\ABP	18	\Afp	18
\Abp	18	\afp	18
\abp	17	\AL	18
\AC	18	\Al	18
\Ac	18	\al	17
\ac	18	\ALP	18
\ACF	18	\Alp	18
\Acf	18	\alp	17
\acf	18	\AnyTrackedLanguages	315, 349
\ACFP	19	\appto	12, 22, 27, 28, 40–45, 74, 79, 93, 101, 131, 133, 169, 179, 181, 182, 351
\Acfp	18	\arabic	23, 390
\acfp	18	\AS	18
\acl	18	\As	18
\acl	18	\as	17
\ACL	18	\ASP	18
\Acl	18	\Asp	18
\acl	18	\asp	17
\ACLP	18	\AtBeginDocument	23, 27, 52, 53, 127
\Aclp	18	\AtEndDocument	45, 94, 102, 121
\aclp	18		
\ACP	18	B	
\Acp	18	babel package	169, 171, 179
\acp	18	\begin	118, 122, 123, 131, 353, 355–362, 381–384, 388
\ACRfullfmt	105	\begingroup	8, 9, 29, 78, 129, 130, 143
\Acrfullfmt	105	\bgroup	35, 111
\acrfullfmt	105	bib2gls	23, 29, 133, 138, 139, 315–317
\ACRfullplfmt	106		
\Acrfullplfmt	105	C	
\acrfullplfmt	105	\c@wrglossary	23
\acronymentry	105	\catcode	47, 125
\acronymfont	70–74, 85, 106	category attributes:	
\acronymname	17	apospplural	183
\acronymsort	105	discardperiod	178
\acronymtype	17, 105, 106	entrycount	91–94, 104
\acrpluralsuffix	105, 127	firstuc	165
\ACS	18	glossdesc	161
\Acs	18	glossdescfont	161
\acs	18	glossname	162
\ACSP	18	glossnamefont	163, 165
\Acsp	18	headuc	299
\acsp	18	indexname	170
\actualchar	172	indexonlyfirst	78
\addtolength	378	insertdots	183
\advance	94, 103, 126, 136	linkcount	143
\AF	18	linkcountmaster	143
\Af	18	markshortwords	183
\af	17	markwords	183, 184, 278, 280, 288
\AFP	18	nohyper	77

inline	363	\gls	34, 50, 51, 92, 104, 109, 122, 138
list	353	\gls@assign@desc	35
listdotted	352	\gls@assign@field	12, 28, 74
listdottedstyle	353	\gls@checkseeallowed	48, 108
listgroup	354	\gls@codepage	121
listhypergroup	354	\gls@defdocnewglossaryentry ..	47, 93, 101
mcolalmtree	383	\gls@defglossaryentry	35, 50, 51
mcolalmtreegroup	383	\gls@dotocitle	111, 112
mcolalmtreehypergroup	384	\gls@glossary	43
mcolalmtreespannav	384	\gls@grplabel	80
mcolindexgroup	380	\gls@level	134
mcolindexhypergroup	380	\gls@noidxglossary	109
mcolindexspannav	380	\gls@org@glossaryentryfield	112
mcoltreegroup	381	\gls@org@glossarysubentryfield ..	112
mcoltreehypergroup	381	\gls@orgTrackLangRequireDialectPrefix	
mcoltreenonamegroup	382	349
mcoltreenonamehypergroup	382	\gls@save@numberlist	53, 54, 56
mcoltreenonamespannav	383	\gls@set@xr@key	43
mcoltreespannav	381	\gls@tmplen	370–379
sublistdotted	353	\gls@type	109
tree	365	\glsabbrvdefaultfont ...	186, 203, 206, 208, 210, 212, 214, 217, 270, 279, 282, 292
treegroup	365	\glsabbrvemfont	
treehypergroup	366	248–257, 259, 260, 262, 264, 266–268
treenoname	366	\glsabbrvfont	84, 85, 106, 186, 190, 191, 193, 194, 196, 199, 203–214, 217, 218, 220, 222, 223, 225, 227, 229, 231, 233, 234, 236, 238, 239, 241, 243, 245, 247, 248, 250, 252, 254, 255, 257, 259, 260, 262, 264, 266, 268, 271, 272, 274, 276–278, 280, 282, 285, 286, 288, 291, 293
treenonamegroup	367	\glsabbrvhyphenfont	279–281, 285, 287–292
treenonamehypergroup	367	\glsabbrvonlyfont	293, 295
glossary-bookindex package	352	\glsabbrvscfont .	219–223, 225, 227, 229–233
glossary-hypernav package	80	\glsabbrvsmfont	234–239, 241, 243–247
glossary-long package	357	\glsabbrvuserfont	270–277
glossary-longbooktabs package	357	\GLSaccessdesc	66
\glossaryentrynumbers	56, 111, 112, 134, 135	\Glsaccessdesc	66, 161, 173
\glossaryheader	118, 131, 353–362, 364–367, 378–382, 384, 388	\glsaccessdesc	66, 162, 177
\glossaryname	111	\GLSaccessdescplural	67
\glossarypostamble	118, 131, 133	\Glsaccessdescplural	67
\glossarypreamble	118, 130	\glsaccessdescplural	67
\glossarysection	118, 124, 130, 133	\GLSaccessfirst	64
\glossarytitle ...	37, 111, 112, 118, 124, 130	\Glsaccessfirst	64
\glossarytoctitle ...	37, 111, 118, 124, 130	\glsaccessfirst	64
\glossentry	112, 135, 352, 353, 355–362, 364–366, 378, 388	\GLSaccessfirstplural	65
\glossentrydesc	352–362, 364–368	\Glsaccessfirstplural	65
\glossentryname		\glsaccessfirstplural	65
....	129, 352–362, 364–366, 378, 379, 386		
\glossentrynameother	130		
\glossentrysymbol	356–360, 362, 364–366, 368		
\glossxtrsetpopts	176		
\GLS	92, 104, 138		
\Gls	50, 92, 104, 138		

<code>\Glsaccesslong</code>	73, 185, 192, 204, 212, 217, 221, 226–229, 235, 241–243, 249, 251, 258, 259, 261, 263– 265, 271–273, 280, 282, 283, 286, 288, 294
<code>\glsaccesslong</code>	72, 73, 185, 192, 193, 203, 206, 208, 209, 211, 212, 214, 215, 217, 218, 220, 222, 224–231, 233, 235–247, 249, 250, 252, 254–267, 269, 271, 272, 275, 277, 278, 280, 282, 283, 286, 288, 293, 294
<code>\Glsaccesslongpl</code>	74, 185, 196, 204, 212, 217, 221, 226– 229, 235, 241–244, 249, 251, 258–261, 263–265, 271–273, 281–283, 286, 289, 294
<code>\glsaccesslongpl</code>	73, 74, 185, 195, 196, 204, 206, 208, 209, 211, 212, 214, 215, 217, 218, 220, 222, 224–233, 235, 237–247, 249, 251, 252, 254, 256–267, 269, 271, 272, 275, 277, 278, 280, 282, 283, 286, 288, 294
<code>\GLSaccessname</code>	66
<code>\Glsaccessname</code>	66
<code>\glsaccessname</code>	41, 66
<code>\GLSaccessplural</code>	65
<code>\Glsaccessplural</code>	64
<code>\glsaccessplural</code>	64
<code>\Glsaccessshort</code>	71, 191, 199, 206, 208–211, 214, 215, 222, 224, 225, 231, 233, 237, 238, 240, 245–247, 252, 254, 256, 257, 267–269, 275, 278, 286, 291
<code>\glsaccessshort</code>	70, 71, 185, 190, 191, 199, 204, 206, 208–212, 214, 215, 217, 220–222, 224–226, 228, 229, 231, 233, 235, 236, 238–243, 245, 247, 249–252, 254–259, 261, 263–269, 271–275, 277, 280, 283, 286, 288, 291, 294
<code>\Glsaccessshortpl</code>	72, 194, 199, 206, 208– 211, 214, 216, 222, 224, 225, 231–233, 237, 238, 240, 245–247, 252, 254, 256, 257, 267–269, 275, 278, 286, 291, 292, 294
<code>\glsaccessshortpl</code>	71, 72, 185, 193, 194, 199, 204, 206, 208–212, 214, 215, 217, 220–222, 224–229, 231, 233, 235, 236, 238–245, 247, 249, 251, 252, 254–261, 263, 265–269, 271, 273–275, 277, 280, 281, 283, 286, 288, 289, 291, 294
<code>\GLSaccesssymbol</code>	67
<code>\Glsaccesssymbol</code>	67, 174
<code>\glsaccesssymbol</code>	67, 173, 178
<code>\GLSaccesssymbolplural</code>	68
<code>\Glsaccesssymbolplural</code>	68
<code>\glsaccesssymbolplural</code>	68
<code>\GLSacesstext</code>	63
<code>\Glsacesstext</code>	63
<code>\glsacesstext</code>	41, 63
<code>\glsacrshortcutstrue</code>	19, 20
<code>\glsacspacemax</code>	107
<code>\glsadd</code>	29, 46, 122
<code>\glsadd options</code>	
<code>theHvalue</code>	9
<code>thevalue</code>	9
<code>\glsaddstoragekey</code>	44, 45, 155, 317
<code>\glsbackslash</code>	49
<code>\glsapscase</code>	58, 62–76, 187–198
<code>\glscategory</code>	56, 63, 76, 84, 85, 156–158, 161– 167, 173, 174, 176, 177, 187–191, 193, 194
<code>\glscategorylabel</code>	76, 180, 182–184, 209, 232, 246, 268, 271, 273, 274, 276, 285, 287, 290, 292
<code>\glsclsebrace</code>	42, 123, 124
<code>\glscounter</code>	9
<code>\glscurrententrylabel</code>	53–55, 112, 120, 129–132, 175, 176
<code>\glscurrentfieldvalue</code>	29–33, 269, 316
<code>\glscustomtext</code> ..	57, 58, 70–74, 187–197, 199
<code>\glsdefaulttype</code>	6, 17, 34, 110, 111, 122, 130, 132
<code>\glsdescriptionaccessdisplay</code> ..	149, 161
<code>\glsdescriptionpluralaccessdisplay</code>	149, 150
<code>\glsdescwidth</code>	355–362
<code>\glsdetoklabel</code>	8, 9, 30– 35, 39–41, 46, 47, 49, 60, 62, 78, 81, 93, 94, 99–103, 108, 112, 114, 115, 129, 130, 134, 137, 139, 161, 163–165, 168, 372, 373
<code>\glsdisplaynumberlist</code>	109, 114
<code>\glsdohyperlink</code>	80, 82
<code>\glsdohypertarget</code>	82, 113
<code>\glsdoifexists</code>	15, 24, 32, 38, 41, 42, 57, 58, 62, 70–74, 82, 108, 115, 116, 129, 130, 187–196
<code>\glsdoifexistsordo</code>	29, 30, 58
<code>\glsdoifexistsorwarn</code>	15, 161, 162, 173
<code>\glsdoifnoexists</code>	35
<code>\glsdonohyperlink</code>	61, 82
<code>\glsdosanitizesort</code>	110
<code>\glsenableentrycount</code>	92, 95, 103

<code>\glsenableentryunitcount</code>	94, 104	<code>\Glsentryuserii</code>	68
<code>\glsentrycounter</code>	23, 119, 120	<code>\glsentryuserii</code>	69
<code>\glsentrycurrcount</code>	93, 94, 101	<code>\Glsentryuseriii</code>	69
<code>\Glsentrydesc</code>	149, 154, 162	<code>\glsentryuseriii</code>	69
<code>\glsentrydesc</code>	149, 153, 154, 162	<code>\Glsentryuseriv</code>	69
<code>\Glsentrydescplural</code>	149, 154	<code>\glsentryuseriv</code>	69
<code>\glsentrydescplural</code>	149, 150, 154	<code>\Glsentryuserv</code>	69
<code>\Glsentryfirst</code>	98, 142, 147, 153	<code>\glsentryuserv</code>	70
<code>\glsentryfirst</code> ...	98, 142, 146, 147, 153, 311	<code>\Glsentryuservi</code>	70
<code>\Glsentryfirstplural</code>	98, 142, 147, 153	<code>\glsentryuservi</code>	70
<code>\glsentryfirstplural</code>	98, 142, 147, 153, 312	<code>\glsextrapostnamehook</code>	166
<code>\glsentryfmt</code>	36–38	<code>\glsfieldfetch</code>	81
<code>\Glsentryfull</code>	106	<code>\glsfieldxdef</code>	160
<code>\glsentryfull</code>	106	<code>\glsfindwidesttoplevelname</code>	370
<code>\Glsentryfullpl</code>	106	<code>\Glsfirst</code>	303
<code>\glsentryfullpl</code>	106	<code>\Glsfirst</code>	304
<code>\glsentryitem</code>	129,	<code>\glsfirst</code>	303
	130, 352, 353, 355–362, 364–366, 378, 389	<code>\glsfirstabbrvdefaultfont</code>	
<code>\Glsentrylong</code>	85, 86, 98, 142, 151, 154		186, 203, 206, 208, 210, 212, 214, 217, 282
<code>\glsentrylong</code> ..	85, 86, 98, 142, 151, 154,	<code>\glsfirstabbrvemfont</code>	248–269
	155, 210, 232, 246, 268, 274, 276, 290, 312	<code>\glsfirstabbrvfont</code> ..	106, 185, 203–206,
<code>\Glsentrylongpl</code>	85, 86, 98, 151, 155		208–217, 220, 222, 223, 225, 227, 229,
<code>\glsentrylongpl</code>	85, 86, 98, 151, 152, 155, 313		231, 233, 234, 236, 238, 239, 241, 243,
<code>\Glsentrylongplural</code>	142		245, 247, 248, 250, 252, 254, 255, 257,
<code>\glsentrylongplural</code>	142		259, 260, 262, 264, 266, 268, 271, 272,
<code>\Glsentryname</code>	145, 152, 163–166		274, 277, 280, 282, 283, 285, 288, 291, 293
<code>\glsentryname</code>	128,	<code>\glsfirstabbrvhyphenfont</code>	
	129, 145, 152, 170, 309, 310, 370–377, 391		279–281, 284, 285, 288–292
<code>\glsentrynumberlist</code>	110, 116, 375–377	<code>\glsfirstabbrvonlyfont</code>	293, 294
<code>\Glsentryplural</code>	146, 152	<code>\glsfirstabbrvscfont</code>	220–233
<code>\glsentryplural</code>	146, 152, 310, 311	<code>\glsfirstabbrvsmfont</code>	234–247
<code>\glsentryprevcount</code>	93, 95, 101	<code>\glsfirstabbrvuserfont</code>	270–278
<code>\glsentryprevmaxcount</code>	101	<code>\glsfirstaccessdisplay</code>	146, 147
<code>\glsentryprevtotalcount</code>	101	<code>\glsfirstlongdefaultfont</code>	
<code>\Glsentryshort</code>	84, 85, 150, 154		203, 206, 212, 214, 217, 220–230, 234–
<code>\glsentryshort</code>	84,		244, 248, 249, 252, 253, 255–260, 262, 263
	85, 107, 150, 154, 272, 273, 284, 308, 309	<code>\glsfirstlongemfont</code>	
<code>\Glsentryshortpl</code>	84, 85, 151, 154		250, 251, 253–255, 260–262, 264, 265
<code>\glsentryshortpl</code>		<code>\glsfirstlongfont</code>	185, 203–208,
	84, 85, 150, 151, 154, 308, 309		210, 212, 214–220, 222, 224, 225, 227,
<code>\Glsentrysymbol</code>	148, 153		229, 231, 233, 235, 236, 238, 239, 241,
<code>\glsentrysymbol</code>	147, 148, 153, 374–376		243, 245, 247, 249, 250, 252, 254, 255,
<code>\Glsentrysymbolplural</code>	148, 153		257, 259, 260, 262, 264, 266, 268, 271,
<code>\glsentrysymbolplural</code>	148, 153		272, 274, 277, 280, 282, 285, 288, 291, 293
<code>\Glsentrytext</code>	145, 152	<code>\glsfirstlongfootnotefont</code>	
<code>\glsentrytext</code>	82, 145, 146, 152, 310		208–211, 230–233, 244–247, 266–269
<code>\glsentrytype</code>	129, 130	<code>\glsfirstlonghyphenfont</code>	279–291
<code>\Glsentryuseri</code>	68	<code>\glsfirstlongonlyfont</code>	293–295
<code>\glsentryuseri</code>	68	<code>\glsfirstlonguserfont</code>	270–278

<code>\GLSfirstplural</code>	304	215, 216, 218–223, 225, 227, 230, 232,
<code>\Glsfirstplural</code>	304	234–239, 241, 244–246, 248–255, 257,
<code>\glsfirstplural</code>	304	259, 260, 262, 264–266, 268, 270–278,
<code>\glsfirstpluralaccessdisplay</code>	147	280–282, 284, 285, 287–290, 292, 293, 295
<code>\glsforeachincategory</code>	200	<code>\glsletentryfield</code>
<code>\glsgenentryfmt</code>	56	<code>\glslink</code>
<code>\glsgetattribute</code>	61,	<code>\glslink options</code>
81, 95, 99–101, 120, 138, 143, 161–167, 169		counter
<code>\glsgetcategoryattribute</code>	156	format
<code>\glsgetgroupitle</code> 354, 355, 364–367, 379–385		hyper
<code>\glsgetwidestname</code>	368	hyperoutside
<code>\glsgroupheading</code>		noindex
.... 134, 353–362, 364–367, 378–384, 390		theHvalue
<code>\glsgroupskip</code> 134, 353, 355–365, 367, 379, 390		thevalue
<code>\glshasattribute</code>		wrgloss
.. 61, 81, 94, 95, 99, 100, 102, 103, 120,		<code>\glslinkcheckfirsthyperhook</code>
138, 143, 161–167, 169, 203, 205, 207,		<code>\glslinkpostsetkeys</code>
208, 210, 213, 215, 216, 218–223, 230,		<code>\glslinkpresetkeys</code>
232, 234–237, 244, 246, 248–255, 262,		<code>\glslinkvar</code>
265, 266, 268, 270, 272–274, 276–278,		<code>\glslistchildpostlocation</code>
280–282, 284, 285, 287–290, 292, 293, 295		<code>\glslistchildprelocation</code>
<code>\glshascategoryattribute</code>	156	<code>\glslistdottedwidth</code>
<code>\glshex</code> .. 320–325, 328–333, 336–338, 341–349		<code>\glslistgroupheaderfmt</code>
<code>\glshyperlink</code>	82	<code>\glslistnavigationitem</code>
<code>\glshypernavsep</code>	117	<code>\glslistprelocation</code>
<code>\glshypernumber</code>	120, 169	<code>\glslocalunset</code>
<code>\glsifattribute</code>	59, 63, 77, 79, 88,	<code>\glslongaccessdisplay</code>
143, 159, 161–164, 168, 175, 178, 299–308		<code>\glslongdefaultfont</code>
<code>\glsifcategory</code>	158 186, 187, 203, 206, 207,
<code>\glsifcategoryattribute</code> . 76, 157, 183, 184		212, 214, 217, 220, 222, 224, 225, 227,
<code>\glsifnotregular</code>	63	229, 235, 236, 238, 239, 241–243, 249,
<code>\glsifnotregularcategory</code>	158	252, 255, 257, 259, 262, 263, 270, 279, 293
<code>\glsifplural</code>		<code>\glslongemfont</code> 248, 250, 253, 254, 260, 261, 264
58, 62, 64, 65, 67, 68, 70–74, 178, 187–198		<code>\glslongfont</code> . 85, 186, 192, 193, 195–197,
<code>\glsifregular</code>	56, 63, 98, 142	203, 204, 206, 208, 210, 212, 214, 216,
<code>\glsifregularcategory</code>	158	217, 219, 220, 222, 224, 225, 227, 229,
<code>\glsifusetranslator</code>	37	231, 233, 235, 236, 238, 239, 241, 243,
<code>\glsignore</code>	54, 55	245, 247, 249, 250, 252, 254, 255, 257,
<code>\glsinlinedescformat</code>	363	259, 260, 262, 264, 266, 268, 271, 272,
<code>\glsinlinesubdescformat</code>	363	274, 277, 280, 282, 285, 288, 291, 293, 294
<code>\glsinsert</code>	58,	<code>\glslongfootnotefont</code>
62, 70–74, 187–198, 279, 285, 287, 290, 292		207, 208, 210, 231, 233, 245, 247, 266, 268
<code>\glskeylisttok</code>	105, 182, 184	<code>\glslonghyphenfont</code>
<code>\glslabel</code>	8, 9, 280, 282, 283, 285, 288, 290, 291
29, 39, 41, 56, 59–61, 76–78, 81, 82, 107,		<code>\glslongonlyfont</code>
139, 143, 177, 178, 197, 198, 210, 232,		293
246, 268, 272–274, 276, 285, 287, 290, 292		<code>\glslongpltok</code>
<code>\glslabeltok</code> 184, 203–205, 207, 208, 216, 219–
. 105, 182, 184, 203–208, 210, 211, 213,		223, 227, 230, 234–237, 241, 244, 248–

253, 255, 259, 260, 264, 266, 270, 271, 273–278, 280–283, 285, 287–289, 293, 295	\glsprestandardsort 110
\glslongpluralaccessdisplay . . . 151, 152	\glsresetentrylist 118, 131
\glslongtok 105, 182–184, 203–209, 211, 213, 216, 218–223, 225, 227, 230, 232, 234–239, 241, 244, 246, 248–255, 257–260, 264, 266, 267, 270, 271, 273–278, 280–285, 287–290, 293–295	\glssee 43–45
\glslonguserfont 270–275, 277	\glsseeformat 41, 42, 47, 109, 115
\glsmcols 381–384	\glsseelist 42
\GLSname 301	\glssetabbrvfmt 56, 63, 84, 85, 161– 167, 173, 174, 187–191, 193, 194, 196, 197
\Glsname 301	\glssetattribute 203, 205, 207, 208, 210, 211, 213, 215, 216, 218–223, 225, 227, 230, 232, 234–239, 241, 244–246, 248–255, 257, 259, 260, 262, 264–266, 268, 271–274, 276–278, 280–282, 284, 285, 287–290, 292, 293, 295
\glsname 301	\glssetcategoryattribute 93, 104, 107, 138, 144, 156, 157, 159, 160, 174
\glsnameaccessdisplay 145, 163–165	\glssetnoexpandfield 12
\glsnamefont 163–166, 168	\glssettoctitle 111
\glsnavhyperlink 117	\glsshortaccessdisplay 150
\glsnavhyperlinkname 80	\glsshortpltok 184, 203–209, 211, 213, 220–223, 225, 230, 232, 234–239, 244, 246, 248– 257, 266, 267, 270, 271, 273, 274, 276– 278, 280, 281, 285, 287–290, 292, 293, 295
\glsnavhypertarget 354, 355, 365–367, 380–385	\glsshortpluralaccessdisplay . . 150, 151
\glsnavigation . . . 354, 355, 365–367, 379–384	\glsshorttok . . 105, 182–184, 203–209, 211, 213, 218, 220, 221, 223, 225, 227, 230, 232, 234–239, 241, 244, 246, 248–256, 258, 260, 266, 267, 270, 271, 273–278, 280, 281, 283, 285, 287–290, 292, 293, 295
\glsnextpages 112	\glssubentryitem 129, 130, 352–362, 364–366, 378, 389
\glsnoidxdisplayloc 115	\glssymbolaccessdisplay 147, 148
\glsnoidxdisplayloclisthandler . . . 114	\glssymbolpluralaccessdisplay 148
\glsnoidxloclist 115, 134, 135	\glstarget 129, 130, 352–362, 364–366, 378, 379, 389, 390
\glsnoidxnumberlistloophandler . . . 115	\GLStext 301, 302
\glsnonnextpages 112	\Glstext 302
\glsnonnumberlistfalse 53	\glstext 301, 302
\glsnonnumberlisttrue 54	\glstextaccessdisplay 145
\glsnopostdotfalse 113	\glstextformat 58, 61
\glsnopostdottrue 112	\glstextup 220
\glsnumberlistloop 110	\glstreechildpredesc 364, 365
\glsnumlistlastsep 114	\glstreechildprelocation . . . 364, 365, 367
\glsnumlistsep 114	\glstreegroupheaderfmt 364–367, 379–385, 387
\glsopenbrace 42, 123, 124	\glstreeindent 365, 366, 377–379
\glsorder 108	\glstreeitem 363, 381, 389
\glspagelistwidth 356, 358, 360, 362	\glstreenamebox 378, 379
\glspar 133	\glstreenamefmt 364–366, 368, 370–379
\GLSpl 92, 104, 138	
\Glspl 51, 92, 104, 138	
\glspl 50, 92, 104, 138	
\GLSplural 302, 303	
\Glsplural 303	
\glsplural 302	
\glspluralaccessdisplay 146	
\glspluralsuffix 127, 183, 187	
\glspostdescription 16, 17, 112, 113, 176, 352–362, 364–368	
\glspostinline 363	
\glspostlinkhook . . . 57, 58, 70–74, 86, 187–196	

<code>\glstreenavigationfmt</code> ..	365–367, 379–384	<code>\glxtr@recordsee</code>	11
<code>\glstreepredesc</code>	364–366	<code>\glxtr@resource</code>	125, 126
<code>\glstreeprelocation</code>	363–366, 368	<code>\glxtr@s@newignoredglossary</code>	36
<code>\glstreesubitem</code>	363, 389	<code>\glxtr@s@provideignoredglossary</code> ...	37
<code>\glstreesubsubitem</code>	364, 390	<code>\glxtr@saveentrycounter</code>	8, 9, 12, 78
<code>\GlsstrLetField</code>	33	<code>\glxtr@setaccessdisplay</code>	167
<code>\glstype</code>	58, 60, 70–74, 139, 187–196	<code>\glxtr@setbookindexmark</code>	391
<code>\glset</code>	46, 58, 95, 96, 139	<code>\glxtr@setup@record</code>	13, 14, 26
<code>\glswrite</code>	42, 108	<code>\glxtr@shortcutsval</code>	126, 127
<code>\glswriteentry</code>	8, 9	<code>\glxtr@texencoding</code>	127
<code>\Glsxtr</code>	51	<code>\glxtr@undefaction@nr</code>	7
<code>\glxtr</code>	51	<code>\glxtr@undefaction@val</code>	7
<code>\glxtr@do@wrglossary</code>	8, 10, 11, 13	<code>\glxtr@usesee</code>	41
<code>\glxtr@addloclistfield</code>	13, 14	<code>\glxtr@warnonexistsordo</code> .	7, 13, 14, 39, 40
<code>\glxtr@addunused</code>	46	<code>\glxtr@writefields</code>	125
<code>\glxtr@applyabbrvfmt</code>	196	<code>\glxtrabbrvfootnote</code>	
<code>\glxtr@applyabbrvstyle</code>	180, 182, 200	208–210, 230–232, 244–246, 266–268
<code>\glxtr@counterrecord</code>	128	<code>\glxtrabbrvpluralsuffix</code>	127,
<code>\glxtr@do@alsoindex@wrglossary</code>	14		187, 203, 205, 208, 210, 212, 214, 216,
<code>\glxtr@doooption</code>	5, 15, 16, 23, 24, 26		220, 234, 248, 270, 280, 282, 285, 291, 293
<code>\glxtr@fields</code>	126	<code>\glxtrabbrvtype</code>	17, 184
<code>\glxtr@headentry@p</code>	88, 89	<code>\glxtractivatenopost</code>	112
<code>\glxtr@hyperoutsidefalse</code>	59	<code>\glxtraddallcrossrefs</code>	45
<code>\glxtr@hyperoutsidettrue</code>	59	<code>\glxtralias</code>	78
<code>\glxtr@ifnextpunc</code>	179	<code>\glxtrAltTreeIndent</code>	368
<code>\glxtr@ifpunctoken</code>	179	<code>\glxtraltrtreeInit</code>	378, 383, 384
<code>\glxtr@inc@linkcount</code>	60, 144	<code>\glxtrAltTreePar</code>	368
<code>\glxtr@inc@wrglossaryctr</code>	8, 9, 23, 26	<code>\glxtrAltTreeSetHangIndent</code> ...	368, 378
<code>\glxtr@indexonly@saveentrycounter</code>		<code>\glxtrAltTreeSetSubHangIndent</code>	379
.....	13, 14, 26	<code>\glxtraltrtreeSubSymbolDescLocation</code>	379
<code>\glxtr@keylist</code>	49–51	<code>\glxtraltrtreeSymbolDescLocation</code> ..	
<code>\glxtr@label</code>	391	368, 378
<code>\glxtr@langtag</code>	127	<code>\glxtrassignfieldfont</code>	63–70
<code>\glxtr@linkprefix</code>	126, 127	<code>\glxtrautoindex</code>	169
<code>\glxtr@loaddialect</code>	315, 349	<code>\glxtrautoindexassignsort</code>	170
<code>\glxtr@makeglossaries</code>	108	<code>\glxtrautoindexentry</code>	170
<code>\glxtr@newabbreviation</code>	106, 182	<code>\glxtrbibaddress</code>	317
<code>\glxtr@next</code>	180	<code>\glxtrbibauthor</code>	317
<code>\glxtr@org@@do@wrglossary</code>	26	<code>\glxtrbibbooktitle</code>	317
<code>\glxtr@org@dohyperlink</code>	80	<code>\glxtrbibchapter</code>	317
<code>\glxtr@org@getgrouptitle</code>	117	<code>\glxtrbibedition</code>	317
<code>\glxtr@org@newignoredglossary</code>	36	<code>\glxtrbibhowpublished</code>	317
<code>\glxtr@org@makenoidxglossaries</code>	47	<code>\glxtrbibinstitution</code>	317
<code>\glxtr@pluralsuffixes</code>	126, 127	<code>\glxtrbibjournal</code>	317
<code>\glxtr@process</code>	131, 132	<code>\glxtrbibmonth</code>	317
<code>\glxtr@provideignoredglossary</code>	37	<code>\glxtrbibnote</code>	317
<code>\glxtr@punctlist</code>	179	<code>\glxtrbibnumber</code>	317
<code>\glxtr@record</code>	10, 11, 126	<code>\glxtrbiborganization</code>	317
<code>\glxtr@record@nr</code>	13	<code>\glxtrbibpages</code>	317

<code>\glxtrbibpublisher</code>	317	<code>\glxtrdisplaysingleloc</code>	118, 119
<code>\glxtrbibschool</code>	317	<code>\glxtrdisplaystartloc</code>	118
<code>\glxtrbibseries</code>	317	<code>\glxtrdoautoindexname</code>	79, 166
<code>\glxtrbibtitle</code>	317	<code>\glxtrdopostpunc</code>	210, 232, 246, 268
<code>\glxtrbibtype</code>	317	<code>\glxtrdowrglossaryhook</code>	79
<code>\glxtrbibvolume</code>	317	<code>\glxtrtremsuffix</code>	248, 250, 252, 254, 255, 257, 259, 260, 262, 264, 266, 268
<code>\glxtrbookindexatendgroup</code>	389	<code>\GlsXtrEnableEntryCounting</code>	104
<code>\glxtrbookindexatsubendgroup</code>	389	<code>\GlsXtrEnableEntryUnitCounting</code>	92
<code>\glxtrbookindexatsubsubendgroup</code> ..	389	<code>\GlsXtrEnableOnTheFly</code>	49, 51, 52
<code>\glxtrbookindexbetween</code>	389	<code>\glxtrtrendfor</code>	31
<code>\glxtrbookindexbookmark</code>	390	<code>\glxtrtrfieldlistgadd</code>	128
<code>\glxtrbookindexcols</code>	388	<code>\glxtrtrfieldtitlecase</code>	161–164, 168
<code>\glxtrbookindexcolspread</code>	388	<code>\glxtrtrfieldtitlecasecs</code>	161
<code>\glxtrbookindexfirstmarkfmt</code>	391	<code>\glxtrtrfieldxifinlist</code>	132
<code>\glxtrbookindexformatheader</code>	390	<code>\glxtrtrfirstscfont</code>	219
<code>\glxtrbookindexgroupskip</code>	390	<code>\glxtrtrfirstsmfont</code>	234
<code>\glxtrbookindexlastmarkfmt</code>	391	<code>\GlsXtrFmtDefaultOptions</code>	29
<code>\glxtrbookindexmulticolseenv</code>	388	<code>\glxtrtrfmtdisplay</code>	29
<code>\glxtrbookindexname</code>	386, 389	<code>\GlsXtrFmtField</code>	29, 30
<code>\glxtrbookindexparentchildsep</code> ..	386, 388	<code>\glxtrtrfootnotename</code> 207, 209, 230, 232, 244, 246, 266, 267
<code>\glxtrbookindexparentsubchildsep</code> .	388, 389	<code>\GlsXtrFormatLocationList</code> ..	53, 56, 375–377
<code>\glxtrbookindexprelocation</code> ...	386, 389	<code>\GLSxtrfull</code>	18, 306, 307
<code>\glxtrbookindexsubbetween</code>	389	<code>\Glsxtrfull</code>	18, 307
<code>\glxtrbookindexsubname</code>	390	<code>\glxtrfull</code>	17, 18, 306, 307
<code>\glxtrbookindexsubprelocation</code> ...	390	<code>\Glsxtrfullformat</code> 186, 198, 201, 204, 206, 208, 210, 212, 214, 218, 220, 222, 224, 226, 228, 230, 231, 233, 235, 237, 239, 240, 242, 244, 245, 247, 249, 251, 252, 254, 256, 257, 260, 262, 263, 265, 266, 268, 271, 272, 275, 278, 280, 283, 286, 288, 291, 294
<code>\glxtrbookindexsubsubbetween</code>	389	<code>\glxtrfullformat</code> 185, 198, 201, 203, 206, 208, 210, 212, 214, 217, 220, 222, 224, 226, 228, 230, 231, 233, 235, 236, 238, 240, 242, 244, 245, 247, 249, 250, 252, 254, 256, 257, 260, 261, 263, 265, 266, 268, 271, 272, 274, 277, 280, 283, 286, 288, 291, 293
<code>\glxtrbookindexthepage</code>	391	<code>\GLSxtrfullpl</code>	18, 19, 307, 308
<code>\glxtrcat</code>	50, 51	<code>\Glsxtrfullpl</code>	18, 308
<code>\glxtrchecknohyperfirst</code>	64, 65	<code>\glxtrfullpl</code>	18, 307
<code>\glxtrcombiningdiacriticIIrules</code> .	321	<code>\Glsxtrfullplformat</code> 186, 198, 201, 204, 206, 208, 210, 212, 214, 218, 221, 222, 224, 226, 228, 230, 231, 233, 235, 237, 239, 240, 242, 244, 245, 247, 249, 251, 252, 254, 256,
<code>\glxtrcombiningdiacriticIIrules</code> ..	321		
<code>\glxtrcombiningdiacriticIrules</code> ...	321		
<code>\glxtrcombiningdiacriticIVrules</code> ..	321		
<code>\glxtrComputeTreeIndent</code>	378		
<code>\glxtrComputeTreeSubIndent</code>	378		
<code>\glxtrcounterprefix</code>	119		
<code>\glxtrcurrencyrules</code>	323		
<code>\Glsxtrdefaultsubsequentfmt</code> ...	199, 201		
<code>\glxtrdefaultsubsequentfmt</code> ...	199, 201		
<code>\Glsxtrdefaultsubsequentplfmt</code> .	199, 201		
<code>\glxtrdefaultsubsequentplfmt</code> .	199, 201		
<code>\GlsXtrDefineAbbreviationShortcuts</code> .	20		
<code>\GlsXtrDefineAcShortcuts</code>	20		
<code>\GlsXtrDefineOtherShortcuts</code>	20		
<code>\glxtrdetoklocation</code>	137		
<code>\glxtrdiscardperiod</code>	177		
<code>\glxtrdisplayendloc</code>	118		
<code>\glxtrdisplayendlochook</code>	119		

257, 260, 262, 263, 265, 267, 268, 271, 272, 275, 278, 281, 283, 286, 289, 291, 294	\glxstrifinmark 62, 88–91, 297, 298
\glxstrfullplformat	\glxstrifnextpunc 179, 180
198, 201, 204, 206, 208, 210, 212, 214, 218, 220, 222, 224, 226, 228, 230, 231, 233, 235, 236, 239, 240, 242, 244, 245, 247, 249, 250, 252, 254, 256, 257, 260, 261, 263, 265, 266, 268, 271, 272, 274, 277, 280, 283, 286, 288, 291, 294	\glxstrifperiod 177, 178
\glxstrfullsep	\glxstrifrecordtrigger 140–142
185, 203–207, 209, 211, 212, 214– 217, 220–229, 231–261, 263–265, 267, 269, 270, 279–281, 283, 284, 288–290, 294	\glxstrifwasfirstuse
\glxstrgenabbrvfmt 56	62, 64, 65, 70–74, 76, 107, 177, 178, 187, 190–196, 209, 210, 232, 246, 268, 272–274, 276, 285, 287, 290, 292
\glxstrgeneralpuncIIrules 324	\glxstrinclinlinkcounter 143
\glxstrgeneralpuncIrules 323	\glxstrindexaliased 77, 78
\glxstrgetgroupitle 117, 390	\glxstrindexseealso 44, 45
\glxstrgroupfield 134	\glxstrinithyperoutside 60
\Glsxtrheadfirst 298	\glxstrinitwrgloss 60, 139
\glxstrheadfirst 298	\glxstrinitwrglossbeforefalse 59
\Glsxtrheadfirstplural 298	\glxstrinitwrglossbeforetrue 59
\glxstrheadfirstplural 298	\Glsxtrinlinefullformat 186, 188, 201, 209, 211, 212, 214, 215, 217, 224– 226, 228, 229, 231, 233, 238, 239, 241– 243, 245, 247, 256–259, 261, 263, 265, 267, 269, 273, 275, 283, 286, 291, 294, 314
\Glsxtrheadfull 298	\glxstrinlinefullformat 186–188, 201, 209, 210, 212, 214, 215, 217, 224–226, 228, 229, 231, 233, 238–240, 242, 243, 245, 247, 255, 257–259, 261, 263, 264, 267, 269, 272, 275, 283, 286, 291, 294, 313
\glxstrheadfull 298	\Glsxtrinlinefullplformat . . 186, 189, 201, 209, 211, 212, 214, 215, 217, 224– 226, 228, 229, 231, 233, 238, 240–242, 244, 246, 247, 256–259, 261, 263, 265, 267, 269, 273, 275, 283, 286, 292, 294, 314
\Glsxtrheadfullpl 298	\glxstrinlinefullplformat
\glxstrheadfullpl 298	186, 189, 190, 201, 209, 211, 212, 214, 215, 217, 224–226, 228, 229, 231, 233, 238–240, 242, 243, 245, 247, 255, 257–259, 261, 263, 264, 267, 269, 272, 275, 283, 286, 291, 294, 314
\Glsxtrheadlong 298	\glxstrinsertinsidefalse 203
\glxstrheadlong 298	\GlsXtrInternalLocationHyperlink 23, 120
\Glsxtrheadlongpl 298	\glxstrLatinA 325–330
\glxstrheadlongpl 298	\glxstrLatinAELigature 327, 329, 330
\Glsxtrheadname 298	\glxstrLatinE 325–330
\glxstrheadname 128, 129, 298	\glxstrLatinEszettSs 326, 328, 331
\Glsxtrheadplural 298	\glxstrLatinEszettSz 327, 329
\glxstrheadplural 298	\glxstrLatinEth 326–329
\Glsxtrheadshort 298	\glxstrLatinH 325–330
\glxstrheadshort 298	\glxstrLatinI 325–330
\Glsxtrheadshortpl 298	\glxstrLatinInsularG 330
\glxstrheadshortpl 298	\glxstrLatinK 326–330
\Glsxtrheadtext 298	\glxstrLatinL 326–330
\glxstrheadtext 298	
\glxstrhyperlink 22, 23, 81	
\glxstrhyphensuffix 280, 288	
\glxstrtrifcounttrigger 95, 96	
\glxstrtrifcustomdiscardperiod 177	
\glxstrtrifemptyglossary 118, 124, 131	
\glxstrtrifhasfield 33, 77, 316, 386	
\glxstrtrifhyphenstart	
279, 281, 284, 287, 289, 290	
\glxstrtrifindexing 78	

<code>\glxtrLatinM</code>	326–330	<code>\glxtrMathItalicSigma</code> .	335, 336, 339, 340
<code>\glxtrLatinN</code>	326–330	<code>\glxtrMathItalicTau</code> ...	335, 336, 339, 340
<code>\glxtrLatinO</code>	326–330	<code>\glxtrMathItalicTheta</code>	335, 339, 340
<code>\glxtrLatinOELigature</code>	328, 330	<code>\glxtrMathItalicUpsilon</code>	335, 336, 339, 340
<code>\glxtrLatinP</code>	326–330	<code>\glxtrMathItalicXi</code>	335, 336, 339, 340
<code>\glxtrLatinS</code>	326–330	<code>\glxtrMathItalicZeta</code>	335, 339, 340
<code>\glxtrLatinT</code>	326–331	<code>\glxtrnewabbrevpresetkeyhook</code>	184
<code>\glxtrLatinThorn</code>	330	<code>\glxtrnewnumber</code>	19
<code>\glxtrLatinX</code>	326–331	<code>\glxtrnewsymbol</code>	19
<code>\glxtrlocationhyperlink</code>	120	<code>\glxtrNoGlossaryWarning</code>	21, 121
<code>\glxtrlocrangefmt</code>	119	<code>\GlsXtrNoGlsWarningAutoMake</code>	124
<code>\GLSxtrlong</code>	18, 305	<code>\GlsXtrNoGlsWarningBuildInfo</code>	125
<code>\Glsxtrlong</code>	18, 306	<code>\GlsXtrNoGlsWarningCheckFile</code>	124
<code>\glxtrlong</code>	17, 18, 305	<code>\GlsXtrNoGlsWarningEmptyMain</code>	124
<code>\glxtrlonghyphen</code>	286	<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	124
<code>\glxtrlonghyphenmashort</code>	282, 283	<code>\GlsXtrNoGlsWarningEmptyStart</code>	124
<code>\glxtrlonghyphenshort</code>	280, 281	<code>\GlsXtrNoGlsWarningHead</code>	124
<code>\glxtrlongnoshortdescname</code> .	216, 264, 282	<code>\GlsXtrNoGlsWarningMisMatch</code>	124
<code>\glxtrlongnoshortname</code>		<code>\GlsXtrNoGlsWarningNoOut</code>	125
.....	218, 227, 241, 258, 260, 283	<code>\GlsXtrNoGlsWarningTail</code>	125
<code>\GLSxtrlongpl</code>	18, 305, 306	<code>\glxtrnopostpunc</code>	112
<code>\Glsxtrlongpl</code>	18, 306	<code>\glxtronlydescname</code>	295
<code>\glxtrlongpl</code>	17, 18, 305	<code>\glxtronlydescsort</code>	295
<code>\glxtrlongshortdescname</code>		<code>\glxtronlyname</code>	293
.....	204, 221, 235, 249, 251, 281, 287	<code>\glxtronlysuffix</code>	293
<code>\glxtrlongshortdescsort</code>		<code>\glxtrorg@ifKV@glslink@hyper</code>	57
....	204, 221, 235, 249, 251, 276, 281, 287	<code>\glxtrorglong</code>	182, 204, 282
<code>\glxtrlongshortname</code>		<code>\glxtrorgshort</code>	182, 204
	203, 220, 234, 248, 250, 270, 271, 280, 285	<code>\GLSxtrp</code>	87
<code>\glxtrlongshortuserdescname</code> ..	273, 276	<code>\Glsxtrp</code>	87
<code>\glxtrmarkhook</code>	296	<code>\glxtrp</code>	87, 89
<code>\glxtrMathItalicAlpha</code>	335, 339, 340	<code>\glxtrparen</code>	177,
<code>\glxtrMathItalicBeta</code>	335, 339, 340		178, 185, 203–207, 209, 211, 212, 214–
<code>\glxtrMathItalicChi</code> ...	335, 336, 339, 340		217, 220–229, 231–261, 263–265, 267,
<code>\glxtrMathItalicDelta</code>	335, 339, 340		269, 270, 279–281, 283, 284, 288–290, 294
<code>\glxtrMathItalicEpsilon</code> ...	335, 339, 340	<code>\Glsxtrpl</code>	51
<code>\glxtrMathItalicEta</code>	335, 339, 340	<code>\glxtrpl</code>	51
<code>\glxtrMathItalicGamma</code>	335, 339, 340	<code>\glxtrpostdescription</code>	
<code>\glxtrMathItalicIota</code>	335, 339, 340	112, 113, 159, 176, 363
<code>\glxtrMathItalicKappa</code> .	335, 336, 339, 340	<code>\glxtrposthyphenlong</code>	290, 292
<code>\glxtrMathItalicLambda</code>	335, 336, 339, 340	<code>\glxtrposthyphenshort</code>	285, 287
<code>\glxtrMathItalicMu</code>	335, 336, 339, 340	<code>\glxtrposthyphensubsequent</code>	
<code>\glxtrMathItalicNu</code>	335, 336, 339, 340	285, 287, 290, 292
<code>\glxtrMathItalicOmega</code> .	335, 336, 340, 341	<code>\glxtrpostlink</code>	177
<code>\glxtrMathItalicOmicron</code>	335, 336, 339, 340	<code>\glxtrpostlinkendsentence</code>	177
<code>\glxtrMathItalicPhi</code> ...	335, 336, 339, 340	<code>\glxtrpostlinkhook</code>	177
<code>\glxtrMathItalicPi</code>	335, 336, 339, 340	<code>\glxtrpostlocalreset</code>	92, 94, 102
<code>\glxtrMathItalicPsi</code> ...	335, 336, 339, 341	<code>\glxtrpostlocalunset</code>	91, 93, 101, 102
<code>\glxtrMathItalicRho</code> ...	335, 336, 339, 340	<code>\glxtrpostlongdescription</code>	35

<code>\glxtrpostnamehook</code>	164–166, 168	<code>\glxtrshortpl</code>	17, 18, 299, 300
<code>\GlsXtrPostNewAbbreviation</code>		<code>\glxtrsmfont</code>	234
.....	184, 200, 201, 203, 205,	<code>\glxtrsmsuffix</code>	
	207–209, 211, 213, 215, 216, 218–223,	234, 236, 238, 239, 241, 243, 245, 247
	225, 227, 230, 232, 234–239, 241, 244,	<code>\Glsxtrsubsequentfmt</code>	
	246, 248–255, 257, 259, 260, 262, 264–	197, 201, 217, 227, 229,
	266, 268, 270, 271, 273, 274, 276–278,		242, 243, 259, 261, 263, 264, 282, 286, 291
	280–282, 284, 285, 287–290, 292, 293, 295	<code>\glxtrsubsequentfmt</code>	
<code>\glxtrpostreset</code>	92, 94, 102	197, 201, 217, 227, 229,
<code>\glxtrpostunset</code>	91, 93, 101		241, 243, 259, 261, 262, 264, 282, 286, 291
<code>\glxtrprelocation</code>		<code>\Glsxtrsubsequentplfmt</code>	
.....	353, 355, 357, 359, 361, 363, 386	197, 201, 217, 227, 229,
<code>\glxtrprotectlinks</code>	81, 82		242, 243, 259, 261, 263, 264, 282, 286, 291
<code>\GlsXtrRecordCounter</code>	11	<code>\glxtrsubsequentplfmt</code>	
<code>\glxtrrecordtriggervalue</code>	138	197, 201, 217, 227, 229,
<code>\glxtrregularfont</code>	56, 63		241, 243, 259, 261, 262, 264, 282, 286, 291
<code>\glxtrresourcecount</code>	126	<code>\glxtrsupplocationurl</code>	120
<code>\glxtrresourcefile</code>	126	<code>\glxtrtagfont</code>	175
<code>\glxtrresourceinit</code>	125	<code>\Glsxtrtitlefirst</code>	297, 298, 311
<code>\glxtrrestoremarkhook</code>	296	<code>\glxtrtitlefirst</code>	297, 298, 311
<code>\glxtrrestorepostpunc</code>	112, 113	<code>\Glsxtrtitlefirstplural</code>	297–299, 312
<code>\glxtrscfont</code>	219	<code>\glxtrtitlefirstplural</code>	297, 298, 312
<code>\glxtrscsuffix</code>		<code>\Glsxtrtitlefull</code>	297–299, 314
....	220, 222, 223, 225, 227, 229, 231, 233	<code>\glxtrtitlefull</code>	297–299, 313
<code>\GlsXtrSetActualChar</code>	172	<code>\Glsxtrtitlefullpl</code>	297–299, 314
<code>\glxtrsetaliasnoindex</code>	13, 14, 78	<code>\glxtrtitlefullpl</code>	297–299, 314
<code>\GlsXtrSetEncapChar</code>	172	<code>\Glsxtrtitlelong</code>	297–299, 312, 313
<code>\GlsXtrSetEscChar</code>	172	<code>\glxtrtitlelong</code>	297–299, 312
<code>\glxtrsetfieldifexists</code>	33	<code>\Glsxtrtitlelongpl</code>	297–299, 313
<code>\GlsXtrSetLevelChar</code>	172	<code>\glxtrtitlelongpl</code>	297–299, 313
<code>\glxtrsetpopts</code>	86	<code>\Glsxtrtitlename</code>	297, 298, 310
<code>\glxtrsetupfulldefs</code>		<code>\glxtrtitlename</code>	297, 298, 309
.....	187–190, 210, 232, 246, 268	<code>\glxtrtitleorpdforheading</code>	
<code>\GLSxtrshort</code>	18, 90, 91, 299, 300	24, 128–130, 297, 298
<code>\Glsxtrshort</code>	18, 300	<code>\Glsxtrtitleplural</code>	297, 298, 311
<code>\glxtrshort</code>	17, 18, 299	<code>\glxtrtitleplural</code>	297, 298, 310, 311
<code>\glxtrshortdescname</code> ...	213, 225, 239, 256	<code>\Glsxtrtitleshort</code>	297, 298, 309
<code>\glxtrshorthyphen</code>	291	<code>\glxtrtitleshort</code>	297, 298, 308
<code>\glxtrshorthyphenlong</code>	288, 289	<code>\Glsxtrtitleshortpl</code>	297, 298, 309
<code>\glxtrshortlongdescname</code>		<code>\glxtrtitleshortpl</code>	297, 298, 308, 309
.....	206, 223, 237, 253, 254, 289, 292	<code>\Glsxtrtitletext</code>	297, 298, 310
<code>\glxtrshortlongdescsort</code>		<code>\glxtrtitletext</code>	297, 298, 310
....	206, 223, 237, 253, 254, 278, 289, 292	<code>\GlsXtrTotalRecordCount</code>	138
<code>\glxtrshortlongname</code>		<code>\glxtrtreetopindent</code>	368, 377
.....	205, 221, 236, 251, 253, 274, 277, 288, 290	<code>\glxtrundefaction</code> ..	7, 13, 14, 27, 36, 39, 40
<code>\glxtrshortlonguserdescname</code> ..	275, 278	<code>\glxtrundeftag</code>	27, 115, 116
<code>\glxtrshortnolongname</code> .	211, 223, 238, 255	<code>\glxtrunsrtdo</code>	132, 133
<code>\GLSxtrshortpl</code>	18, 299, 300	<code>\glxtrUpAlpha</code>	334, 339, 340
<code>\Glsxtrshortpl</code>	18, 300	<code>\glxtrUpBeta</code>	334, 339, 340

`\ifglsnopostdot` 16, 112
`\ifglssanitizesort` 110
`\ifglssubentrycounter` 34
`\ifglssused` 45, 46, 76,
78, 79, 94, 103, 107, 197, 370–372, 374–376
`\ifglsxindy` 121–123
`\ifglxtr@hyperoutside` 61
`\ifglxtrinitwrglossbefore` ... 59, 61, 139
`\ifglxtrininsertinside`
..... 190–196, 199, 203, 204, 206,
208–212, 214–218, 220–222, 224–233,
235–247, 249–252, 254–269, 271–275,
277–279, 281, 282, 284–286, 288, 290–294
`\ifHy@hyperindex` 169
`\ifinlistcs` 31, 47
`\ifinner` 24
`\ifKV@glslink@hyper` 57, 60, 61
`\ifKV@glslink@local` 58, 139
`\ifKV@glslink@noindex` 8, 9, 12, 29, 78
`\ifmmode` 24
`\ifnum` 14,
15, 46, 95, 103, 116, 126, 138, 365, 366, 378
`\ifstrempy` 129, 136, 144
`\ifstrequal` 16, 21
`\ifthenelse` 124
`\IfTrackedLanguageFileExists` 315
`\ifundef` 23, 32, 108, 174–176, 349
`\ifx` 8–11,
42, 52, 53, 107, 108, 111, 112, 118–120,
127, 169–171, 173, 179–182, 184, 279, 385
`\immediate` 94, 103, 121, 127
`\index` 170
`\indexspace` . 353, 364–367, 379–385, 387, 390
`\input` 315
`\inputencodingname` 127
`\InputIfFileExists` 47
`\istfilename` 108
`\item` 122, 123, 352–355, 363–365, 380, 381

J

`\jobname` 47, 120–127

K

`\key@ifundefined` 12, 27, 28, 74, 131, 134
`\KV@glslink@hyperfalse` .. 63, 76, 77, 82, 83
`\KV@glslink@hypertrue` 82
`\KV@glslink@noindexfalse` 77
`\KV@glslink@noindextrue` 77, 83

L

`\L` 330, 333
`\l` 333
`\label` 23
`\LaTeX` 122, 123
`\leaders` 352
`\leavevmode` 36, 60
`\let` 5, 7–11, 13–
15, 17–19, 24–26, 29, 31, 35, 47, 48, 52,
54, 55, 57, 58, 60–80, 82, 83, 86, 92–94,
101–117, 125, 127, 131, 132, 134, 139,
144, 161–171, 174–176, 179–183, 187–
196, 199, 201, 210, 232, 246, 268, 296–
299, 349, 363, 364, 368, 370, 381, 388–391
`\letabbreviationstyle` .. 209, 211, 213,
215, 218, 219, 225, 226, 239, 240, 256, 258
`\letcs` 28, 32, 41,
46, 61, 74, 114–117, 133, 134, 161–168, 391
`\levelchar` 172
`\listadd` 99
`\listbreak` 175
`\listcsadd` 30
`\listcseadd` 30, 100
`\listcsgadd` 30, 47
`\listcsxadd` 31, 99
`\loadglsentries` 48, 122
`\long` 35

M

`\MakeAcronymsAbbreviations` 106
`\makeatletter` 47, 120, 125, 171
`\makeatother` 171
`\makebox` 352, 378, 379
`\makefirstuc` 175
`makeglossaries` 114
`\makeglossaries` 107, 121, 123, 124, 128
`\makeglossary` 108
`makeindex` 392
`makeindex` 14, 107
`\makenoidxglossaries` 123
`\MakeTextUppercase` 298
`\MakeUppercase` 297, 298
`\marginpar` 24
`\markboth` 296
`\markright` 296
`\mathit` 318
`\mathrm` 317–319
`\maxdimen` 52, 53
`\mbox` 354, 355, 378

<code>\medskip</code>	124, 133	<code>group</code>	133, 134
<code>\MessageBreak</code>		<code>loclist</code>	30
... 48, 51, 95, 104, 107, 108, 110, 111, 200		<code>long</code>	151, 155, 312
<code>mfirstuc</code> package	174, 175	<code>longplural</code>	152, 155, 313
<code>\mfirstucMakeUppercase</code>		<code>name</code> ... 41, 144, 145, 152, 169, 300, 301, 309	
..... 63–74, 76, 84–86, 88, 90,		<code>plural</code>	146, 152, 202, 302, 303, 310
91, 97, 98, 105, 106, 142, 145–155, 163,		<code>see</code>	15, 25, 40, 41, 43, 46, 48, 108
165, 168, 188, 190, 191, 193, 194, 196–198		<code>seealso</code>	15, 40, 41, 43, 45, 46, 403
<code>\mfu@checkword@arg</code>	175	<code>short</code>	150, 154, 181
<code>\mfu@checkword@do</code>	175	<code>shortplural</code>	151, 154, 181
N			
<code>\NeedsTeXFormat</code>	5, 316, 351, 386	<code>symbol</code>	147, 148, 153
<code>\new@glossaryentry</code>	48, 110	<code>symbolplural</code>	148, 153
<code>\new@ifnextchar</code>		<code>text</code> ... 80, 145, 152, 202, 204, 301, 302, 310	
29, 75, 76, 97, 136, 139–141, 179, 187–196		<code>\newglossarystyle</code>	388
<code>\newabbr</code>	18, 19	<code>\newif</code>	59, 168, 203
<code>\newabbreviation</code>	18, 19	<code>\newlength</code>	368
<code>\newabbreviationhook</code>	184	<code>\newnum</code>	19
<code>\newabbreviationstyle</code>	203–	<code>\newrobustcmd</code>	29–33, 42,
207, 209, 211, 213, 215, 216, 218–221,		43, 75, 76, 86–88, 97, 112, 116, 128, 129,	
223, 225–228, 230, 232, 234–237, 239–		132, 133, 136, 137, 139–141, 167, 174,	
241, 243, 244, 246, 248–251, 253–256,		175, 187–197, 279, 297, 299–308, 370–376	
258, 260, 262, 264–267, 270, 271, 273–		<code>\newsym</code>	19
278, 280–283, 285, 287–290, 292, 293, 295		<code>\newterm</code>	159
<code>\newacronym</code>	105, 106	<code>\newtoks</code>	181
<code>\newacronymhook</code>	105	<code>\newwrite</code>	108
<code>\newacronymstyle</code>	106	<code>\nobreak</code>	354, 355, 364–367, 380–383
<code>\newcommand</code>	5–32, 34–38, 40–	<code>\NoCaseChange</code>	88–91, 129, 299–308
42, 44–46, 48–52, 54–56, 59, 60, 62, 63,		<code>\noexpand</code>	10,
74–83, 86, 88–93, 95, 97–101, 103, 104,		11, 21, 42, 44, 45, 47, 105, 121, 125,	
106, 107, 111–130, 132–161, 166, 167,		131, 132, 134, 143, 170, 171, 184, 351, 389	
169–182, 185–197, 199–207, 211, 213,		<code>\nofiles</code>	124
216, 218–220, 234, 248, 269, 270, 273,		<code>\noindent</code>	124, 366, 367, 381–384
275, 279–281, 284, 285, 287, 289, 290,		<code>\nopagebreak</code>	364–367, 379–386, 390
292–294, 296–317, 320–348, 351, 353,		<code>\nopostdesc</code>	36, 50, 51, 112, 159
363, 367–370, 377, 378, 386, 387, 390, 391		<code>\ns@GLSxtrfull</code>	188
<code>\newcount</code>	125, 135	<code>\ns@Glsxtrfull</code>	187
<code>\newcounter</code>	23, 143	<code>\ns@glxtrfull</code>	187
<code>\newentry</code>	19	<code>\ns@GLSxtrfullpl</code>	189
<code>\newglossary</code>	17, 108	<code>\ns@Glsxtrfullpl</code>	189
<code>\newglossaryentry</code>		<code>\ns@glxtrfullpl</code>	188
..... 19, 48, 93, 101, 105, 159, 160, 184		<code>\ns@GLSxtrlong</code>	192, 193
<code>\newglossaryentry options</code>		<code>\ns@Glsxtrlong</code>	192
alias	15, 40, 43–46	<code>\ns@glxtrlong</code>	191, 192
desc	149, 153, 154	<code>\ns@glxtrlongpl</code>	196
descplural	149, 154	<code>\ns@Glsxtrlongpl</code>	195
first 80, 146, 147, 153, 202, 303, 304, 311, 392		<code>\ns@glxtrlongpl</code>	195
firstplural	147, 153, 202, 304, 311, 392	<code>\ns@GLSxtrshort</code>	191
		<code>\ns@Glsxtrshort</code>	190
		<code>\ns@glxtrshort</code>	190

<code>\ns@GLSxtrshortpl</code>	194	<code>sort</code>	
<code>\ns@Glsxtrshortpl</code>	194	<code>use</code>	62
<code>\ns@glxtrshortpl</code>	193	<code>style</code>	22
<code>\null</code>	21	<code>stylemods</code>	22
<code>\number</code>	47, 100–102, 125, 136	<code>symbols</code>	19, 159
<code>\numexpr</code>	100, 102	<code>undefaction</code>	39
O			
<code>\O</code>	330, 333	<code>error</code>	6
<code>\o</code>	333	<code>warn</code>	6
<code>\or</code>	7, 13, 14, 19, 20, 23, 24, 48, 59, 364, 389	<code>xindy</code>	42, 43
<code>\org@glossaryentrynumbers</code>	53, 112	<code>\PackageError</code>	
<code>\org@glossarytitle</code>	111, 112	6, 11, 21, 25, 48, 51, 52, 75, 76, 78,
<code>\org@ifKV@glslink@hyper</code>	60, 61	87, 92–94, 101, 103, 104, 106–108, 110,
P			
<code>\p@gl@hyp@opt</code>	79	117, 127, 132, 133, 135, 199–202, 351, 352
package options:		<code>\PackageWarning</code>	15
<code>abbreviations</code>	17	<code>\PackageWarningNoLine</code>	15
<code>accsupp</code>	20, 144	<code>\pageref</code>	22, 34
<code>acronym</code>	17	<code>\par</code> .	124, 125, 354, 355, 364–368, 378–384, 387
<code>automake</code>	111, 122, 127	<code>\parindent</code>	
<code>true</code>	127	363, 365, 366, 368, 378, 379, 381–385, 388
<code>autoseeindex</code>	25	<code>\parskip</code>	363, 365, 366, 381–383, 388
<code>false</code>	25	<code>\PassOptionsToPackage</code>	5, 21
<code>counter</code>		<code>\pdfbookmark</code>	387
<code>wrglossary</code>	23	<code>\preglossarypreamble</code>	34
<code>debug</code>		<code>\preto</code>	77
<code>showtargets</code>	81	<code>\print@noop@unsrtglossaryunit</code> ...	11, 13
<code>docdef</code>	14, 47, 48, 93, 101	<code>\print@op@unsrtglossaryunit</code>	13, 14
<code>false</code>	47, 48	<code>\printabbreviations</code>	17
<code>restricted</code>	15	<code>\printglossaries</code>	109, 123
<code>true</code>	46, 48	<code>\printglossary</code>	17, 109, 123
<code>docdefs</code>		<code>\printglossary options</code>	
<code>restricted</code>	47	<code>nonumberlist</code>	55
<code>indexcounter</code>	316	<code>type</code>	110
<code>nonumberlist</code>	53	<code>\printnoidxglossaries</code>	123
<code>nopostdot</code>	16	<code>\printnoidxglossary</code>	109, 123
<code>false</code>	16	<code>\printnumbers</code>	19, 160
<code>numbers</code>	19	<code>\printsymbols</code>	19, 159
<code>postdot</code>	16	<code>\printunsrtglossary</code>	130
<code>record</code>	7, 12, 13, 47, 57, 107, 125, 401	<code>\printunsrtglossaryentryprocesshook</code>	131
<code>alsoindex</code>	8, 10	<code>\printunsrtglossaryhandler</code>	132
<code>only</code>	8	<code>\printunsrtglossarypredoglossary</code> ..	131
<code>shortcuts</code>	19	<code>\printunsrtglossaryskipentry</code> ..	131, 132
<code>ac</code>	19	<code>\printunsrtglossaryunit</code>	13, 14, 133
<code>all</code>	19	<code>\printunsrtglossaryunitsetup</code>	132
<code>false</code>	19	<code>\ProcessOptions</code>	352
<code>none</code>	19	<code>\ProcessOptionsX</code>	24
<code>true</code>	19	<code>\protect</code>	88–91, 152–155, 182, 185,
		203–213, 215–268, 270, 271, 273–278,
			280–283, 285, 287–290, 292–295, 299–308
		<code>\protected@csedef</code>	32, 33, 117, 368, 369

<code>\textsmaller</code>	234	<code>\unskip</code>	36, 46, 352
<code>\texttt</code>	24, 121–124	<code>upgreek</code> package	318
<code>\the</code>	105, 119, 120, 126, 173, 184, 203–211, 213, 215, 216, 218–223, 225, 227, 230, 232, 234– 239, 241, 244–246, 248–260, 262, 264– 268, 270–278, 280–285, 287–290, 292–295	<code>\usepackage</code>	123, 124
<code>\theglsentrycounter</code>	8–11, 60, 62, 139	W	
<code>\theHglentrycounter</code>	8–11, 60, 62, 139	<code>\warn@nomakeglossaries</code>	109
<code>\theindex</code>	169	<code>\warn@noprintglossary</code>	109, 112
<code>\thewrglossary</code>	23	<code>wrglossary</code> (counter)	22, 23
<code>\this@dialect</code>	315, 349	<code>\write</code>	42, 94, 103, 108, 121, 127
<code>\thisgrptitle</code>	390	X	
<code>\toks@</code>	119, 120, 173	<code>\x</code>	120, 143
<code>tracklang</code> package	127, 319	<code>\xcapitalisewords</code>	161
<code>\TrackLangGetDefaultScript</code>	349	<code>\xdef</code>	112, 132
<code>\TrackLangIfHasDefaultScript</code>	349	<code>\xifinlist</code>	99
<code>\TrackLangRequireDialectPrefix</code>	349	<code>\xifinlistcs</code>	31
U		<code>xindy</code>	392
<code>\u</code>	316	<code>xindy</code>	14, 107
<code>\undef</code>	13, 14, 47, 174	<code>xkeyval</code> package	5
<code>\underline</code>	175	<code>\XKV@checkchoice</code>	56
		<code>\XKV@plfalse</code>	55
		<code>\XKV@resa</code>	56
		<code>\XKV@sttrue</code>	55