

`glossaries-extra.sty v1.43: documented code`

Nicola L.C. Talbot

Dickimaw Books

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

2020-02-28

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	32
1.3	Modifications to Commands Provided by glossaries	47
1.3.1	Existence Checks	52
1.3.2	Document Definitions	62
1.3.3	Existing Glossary Style Modifications	68
1.3.4	Entry Formatting, Hyperlinks and Indexing	72
1.3.5	Entry Counting	111
1.3.6	Acronym Modifications	127
1.3.7	Indexing and Displaying Glossaries	131
1.4	Link Counting	170
1.5	Integration with glossaries-accsupp	171
1.6	Categories	189
1.7	Abbreviations	216
1.7.1	Abbreviation Styles Setup	237
1.7.2	Predefined Styles (Default Font)	240
1.7.3	Predefined Styles (Small Capitals)	259
1.7.4	Predefined Styles (Fake Small Capitals)	276
1.7.5	Predefined Styles (Emphasized)	293
1.7.6	Predefined Styles (User Parentheses Hook)	317
1.7.7	Predefined Styles (Hyphen)	328
1.7.8	Predefined Styles (No Short on First Use)	343
1.8	Using Entries in Headings	346
1.9	Multi-Lingual Support	369
1.10	glossaries-extra-bib2gls.sty	370
2	Style Adjustments (glossaries-extra-stylemods.sty)	416
2.1	Package Initialisation	416
2.2	List-Like Styles	417
2.3	Longtable Styles	420
2.4	Long Ragged Styles	422
2.5	Supertabular Styles	424
2.6	Super Ragged Styles	426
2.7	Inline Style	428
2.8	Tree Styles	428
2.9	Multicolumn Styles	448

3	bookindex style (glossary-bookindex.sty)	454
3.1	Package Initialisation and Options	454
4	longextra styles (glossary-longextra.sty)	461
4.1	Package Initialisation and Options	461
5	topic styles (glossary-topic.sty)	484
5.1	Package Initialisation and Options	484
	Glossary	490
	Change History	491
	Index	516

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2020/02/28 v1.43 (NLCT)]

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

Requires etoolbox package.
4 \RequirePackage{etoolbox}

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

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

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

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

Declare package options.

`\glxtrundefaction` 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*{\glxtrundefaction}[2]{%
30   \@glxtrundeftag\PackageError{glossaries-extra}{#1}{#2}%
31 }

```

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

```

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

```

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

```

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

```

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 `\glxtrundefaction` should behave if `undefaction=warn` is set.

```

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

```

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

```

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

```

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

```

41 \newcommand*{\@glxtr@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*{\@glxtr@redef@for@gl@sentries}{}

```

`\f@for@gl@sentries`

```

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

```

```

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

```

undefaction

```

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

```

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

\glstr@record Does nothing by default.

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

```

\glstr@recordsee Does nothing by default.

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

```

\glstr@defaultnumberformat

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

```

\GlsXtrSetDefaultNumberFormat

```

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

```

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

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

```

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

```

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

```

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

```

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

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

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

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

Save the label in case it's needed. This needs to be outside the existence check to allow the post-link hook to reference it.

```
121 \edef\@gls@label{\glsdetoklabel{#2}}%
122 \let\glslabel\@gls@label
123 \ifglsentryexists{#2}{}%
124 {%
125   \@@glsxtrwrglossmark
126   \begingroup
127     \let\@glsnumberformat\@glsxtr@defaultnumberformat
128     \def\@glsxtr@thevalue{}%
129     \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
130     \let\@glsxtr@org@theHvalue\@glsxtr@theHvalue
```

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

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

Unless the equations option is on and this is inside a numbered maths environment.

```
132     \ifglsxtr@equations
133       \@glsxtr@use@equation@counter
134     \fi
```

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

```
135     \@gls@setdefault@glslink@opts
```

Implement any pre-key settings.

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

Assign keys.

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

Implement any post-key settings. Is the auto-add on?

```
138     \glsxtr@do@autoadd{#3}%
```

Check post-key hook.

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

Increment associated counter.

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

Check if noindex option has been used.

```
141     \ifKV@glslink@noindex
142     \else
143       \glswriteentry{#2}%
144     {%
```

Check if thevalue has been set.

```
145       \ifdefempty{\@glsxtr@thevalue}%
146       {%
```

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

```

147         \ifx\@glxtr@org@theHvalue\@glxtr@theHvalue
148         \else
149         \let\theHglentrycounter\@glxtr@theHvalue
150         \fi

```

Save the entry counter.

```

151         \glxtr@saveentrycounter

```

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

```

152         \let\@@do@@wrglossary\@glxtr@dorecord
153     }%
154     {%

```

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.)

```

155         \let\theHglentrycounter\@glxtr@thevalue
156         \let\theHglentrycounter\@glxtr@theHvalue
157         \let\@@do@@wrglossary\@glxtr@dorecordnodefer
158     }%
159     \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
160     \glxtr@@do@@wrglossary{#2}%
161     \else

```

No need to escape special characters.

```

162         \@@do@@wrglossary
163     \fi
164 }%
165 \fi
166 \endgroup
167 }%
168 }

```

glslink@prekeys

```

169 \newcommand{\@glxtr@glslink@prekeys}{\glslinkpresetkeys}

```

lslink@postkeys

```

170 \newcommand{\@glxtr@glslink@postkeys}{\glslinkpostsetkeys}

```

lossadd@prekeys

```

171 \newcommand{\@glxtr@glossadd@prekeys}{\glsaddpresetkeys}

```

ossadd@postkeys

```

172 \newcommand{\@glxtr@glossadd@postkeys}{\glsaddpostsetkeys}

```

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

```

173 \newcommand*\@glxtr@dorecord{%

```

```

174 \global\let\@glsrecordlocref\theglsentrycounter
175 \let\@glsxtr@orgprefix\@glo@counterprefix
176 \ifx\theglsentrycounter\theHglentrycounter
177   \def\@glo@counterprefix{}%
178 \else
  Protect against non-expandable commands occurring in the location.
179   \protected@edef\@glsxtr@theentrycounter{\theglsentrycounter}%
180   \protected@edef\@glsxtr@theHentrycounter{\theHglentrycounter}%
181   \@onelevel@sanitize\@glsxtr@theentrycounter
182   \@onelevel@sanitize\@glsxtr@theHentrycounter
183   \protected@edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
184     {\@glsxtr@theentrycounter}{\@glsxtr@theHentrycounter}%
185   }%
186   \@do@gls@getcounterprefix
187 \fi

```

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

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

```

188 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
189   \@glsxtr@do@nameref@record
190   {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
191   {\@glsrecordlocref}%
192 \else
193   \protected@write\@auxout{}{\string\@glsxtr@record
194     {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
195     {\@glsrecordlocref}}%
196 \fi
197 \@glsxtr@counterrecordhook
198 \let\@glo@counterprefix\@glsxtr@orgprefix
199 }

```

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

```

200 \newcommand*\@glsxtr@dorecordnodefer{%
201   \ifx\theglsentrycounter\theHglentrycounter
202     \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
203       \@glsxtr@do@nameref@record
204       {\@gls@label}{\@gls@counter}{\@glsnumberformat}%
205       {\theglsentrycounter}%
206     \else
207       \protected@write\@auxout{}{\string\@glsxtr@record
208         {\@gls@label}{\@gls@counter}{\@glsnumberformat}%
209         {\theglsentrycounter}}%
210     \fi
211   \else
212     \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix

```

```

213     {\theglsentrycounter}{\theHglentrycounter}%
214 }%
215 \@do@gls@getcounterprefix
216 \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
217   \@glxtr@do@nameref@record
218     {\@gls@label}{\@glo@counterprefix}{\@gls@counter}%
219     {\@gls@numberformat}{\theglsentrycounter}%
220 \else
221   \protected@write\@auxout{}\string\glxtr@record
222     {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@gls@numberformat}%
223     {\theglsentrycounter}}%
224 \fi
225 \fi
226 \@glxtr@counterrecordhook
227 }

```

`\glxtr@ifnum@mmode` Check if in a numbered maths environment. The `amsmath` package is automatically loaded by `datatool-base`, which is required by `glossaries`, so `\ifst@rred` and `\if@display` should both be defined.

```

228 \newcommand{\@glxtr@ifnum@mmode}[2]{%
229   \ifmmode
230     \ifst@rred
231       #2%
232   \else

```

Non-`amsmath` environments and regular inline math mode isn't flagged as starred by `amsmath`, but we can't use `\mathchoice` in this case as it's not the current style that's relevant. Instead we can use `amsmath's \if@display`. This may not work for environments that aren't provided by `amsmath`.

```

233     \if@display #1\else #2\fi
234   \fi
235 \else
236   #2%
237 \fi
238 }

```

`\glxtr@nameref@record` With `record=nameref`, the current label information is included in the record, but this may not have been defined, so `\csuse` will prevent an undefined control sequence error and just leave the last two arguments blank if there's no information. In the event that a record is in `amsmath's align` environment `\@currentHref` will be out. There may be other instances where `\@currentHref` is out, so this also saves `\theHglentrycounter`, which is useful if it can't be obtained by prefixing `\theglsentrycounter`.

```

239 \newcommand*{\@glxtr@do@nameref@record}[5]{%
240   \gls@ifnotmeasuring
241   {%
242     \protected@write\@auxout{}\string\glxtr@record@nameref
243       {#1}{#2}{#3}{#4}{#5}%
244     {\csuse{@currentlabelname}}{\csuse{@currentHref}}}%

```

```

245     {\theHglentrycounter}}%
246   }%
247 }

r@recordcounter
248 \newcommand*{\@glxtr@recordcounter}{%
249   \@glxtr@noop@recordcounter
250 }

p@recordcounter
251 \newcommand*{\@glxtr@noop@recordcounter}[1]{%
252   \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
253     requires record=only or record=alsoindex package option}{}%
254 }

p@recordcounter
255 \newcommand*{\@glxtr@op@recordcounter}[1]{%
256   \eappto\@glxtr@counterrecordhook{\noexpand\@glxtr@docounterrecord{#1}}%
257 }

lsxtr@recordsee Deal with \glssee in record mode. (This doesn't increment the associated counter.)
258 \newcommand*{\@glxtr@recordsee}[2]{%
259   \@glxtr@wrglossmark
260   \def\@gls@xref{#2}%
261   \@onelevel@sanitize\@gls@xref
262   \protected@write\@auxout{}\{\string\glxtr@recordsee{#1}-{\@gls@xref}}%
263 }

srtglossaryunit
264 \newcommand{\printunsrtglossaryunit}{%
265   \print@noop@unsrtglossaryunit
266 }

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

saveentrycounter Only store the entry counter information if the indexing is on.
268 \newcommand*{\glxtr@indexonly@saveentrycounter}{%
269   \ifKV@glslink@noindex
270   \else
271     \glxtr@saveentrycounter
272   \fi
273 }

addloclistfield
274 \newcommand*{\glxtr@addloclistfield}{%
275   \key@ifundefined{glossentry}{loclist}%
276   {%

```

```

277 \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
278 \appto\@gls@keymap{,{loclist}{loclist}}%
279 \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
280 \appto\@newglossaryentryposthook{%
281   \gls@assign@field{\@glo@label}{loclist}{\@glo@loclist}%
282 }%
283 \glssetnoexpandfield{loclist}%
284 }%
285 {}%

```

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

```

286 \key@ifundefined{glossentry}{location}%
287 {%
288   \define@key{glossentry}{location}{\def\@glo@location{##1}}%
289   \appto\@gls@keymap{,{location}{location}}%
290   \appto\@newglossaryentryprehook{\def\@glo@location{}}%
291   \appto\@newglossaryentryposthook{%
292     \gls@assign@field{\@glo@label}{location}{\@glo@location}%
293   }%
294   \glssetnoexpandfield{location}%
295 }%
296 {}%

```

Add a key to store the group heading.

```

297 \key@ifundefined{glossentry}{group}%
298 {%
299   \define@key{glossentry}{group}{\def\@glo@group{##1}}%
300   \appto\@gls@keymap{,{group}{group}}%
301   \appto\@newglossaryentryprehook{\def\@glo@group{}}%
302   \appto\@newglossaryentryposthook{%
303     \gls@assign@field{\@glo@label}{group}{\@glo@group}%
304   }%
305   \glssetnoexpandfield{group}%
306 }%
307 {}%
308 }

```

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

```

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

```

`ttting@alsoindex`

```

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

```

`rd@setting@only`

```

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

```

`setting@nameref`

```

312 \newcommand*{\@glsxtr@record@setting@nameref}{nameref}

```

@if@record@only

```
313 \newcommand*{\@glxtr@if@record@only}[2]{%
314   \ifx\@glxtr@record@setting\@glxtr@record@setting@only
315     #1%
316   \else
317     \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
318       #1%
319     \else
320       #2%
321     \fi
322   \fi
323 }
```

ord@setting@off

```
324 \newcommand*{\@glxtr@record@setting@off}{off}
```

cord@only@setup Initialisation code for record=only and record=nameref

```
325 \newcommand*{\@glxtr@record@only@setup}{%
326   \def\glxtr@setup@record{%
327     \@glxtr@autoseeindexfalse
328     \let\@do@seeglossary\@glxtr@recordsee
329     \let\@glxtr@record\@glxtr@record
330     \let\@do@wrglossary\@glxtr@do@record@wrglossary
331     \let\@gls@saveentrycounter\relax
332     \let\glxtrundefaction\@glxtr@warn@undefaction
333     \let\glxtr@warnonexistsordo\@glxtr@warn@onexistsordo
334     \glxtr@addloclistfield
335     \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
336     \let\@glxtr@recordcounter\@glxtr@op@recordcounter
337     \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
```

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

```
338   \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.

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

Warn about using \printglossary:

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

Load glossaries-extra-bib2gls:

```
341   \RequirePackage{glossaries-extra-bib2gls}%
342 }%
343 }
```

record Now define the record package option.

```
344 \define@choicekey{glossaries-extra.sty}{record}
345 [ \@glxtr@record@setting\glxtr@record@nr ]%
```

```

346 {off,only,alsoindex,nameref}%
347 [only]%
348 {%
349 \ifcase\glxtr@record@nr\relax

```

Don't record.

```

350 \def\glxtr@setup@record{%
351 \renewcommand*{\@do@seeglossary}{\@glxtr@doseeglossary}%
352 \renewcommand*{\@glxtr@record}[3]{}%
353 \let\@do@wrglossary\glxtr@do@wrglossary
354 \let\@glxtr@saveentrycounter\glxtr@indexonly@saveentrycounter
355 \let\glxtrundefaction\@glxtr@err@undefaction
356 \let\glxtr@warnonexistsordo\@gobble
357 \let\@glxtr@recordcounter\@glxtr@noop@recordcounter
358 \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
359 \undef\glxtrsetaliasnoindex
360 }%
361 \or

```

Only record (don't index).

```

362 \@glxtr@record@only@setup
363 \or

```

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

```

364 \def\glxtr@setup@record{%
365 \renewcommand*{\@do@seeglossary}{\@glxtr@dosee@alsoindex@glossary}%
366 \let\@glxtr@record\@glxtr@record
367 \let\@do@wrglossary\glxtr@do@alsoindex@wrglossary
368 \let\@glxtr@saveentrycounter\glxtr@indexonly@saveentrycounter
369 \let\glxtrundefaction\@glxtr@warn@undefaction
370 \let\glxtr@warnonexistsordo\@glxtr@warn@onexistsordo
371 \glxtr@addloclistfield
372 \let\@glxtr@recordcounter\@glxtr@op@recordcounter
373 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
374 \undef\glxtrsetaliasnoindex
375 }%
376 \or

```

Only record (don't index) but also include nameref information.

```

377 \@glxtr@record@only@setup
378 \ifundef\hyperlink
379 {\GlossariesExtraWarning{You have requested record=nameref but
380 the document doesn't support hyperlinks}}%
381 }%
382 \fi
383 }

```

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

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

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

Need to provide conditional commands that are backward compatible:

`if@glxtrdocdef`

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

`lsxtrdocdeftrue`

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

`sxtrdocdeffalse`

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

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

```
388 \define@choicekey{glossaries-extra.sty}{docdef}
389 [\@glxtr@docdefsetting\@glxtr@docdefval]%
390 {false,true,restricted,atom}[true]%
391 {%
392   \ifnum\@glxtr@docdefval>1\relax
393     \renewcommand*{\@glsdofexistsorwarn}{\glsdofexists}%
394   \else
395     \renewcommand*{\@glsdofexistsorwarn}{\glsdofexistsorwarn}%
396   \fi
397 }
```

`ocdefrestricted`

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

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

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

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

```
400 \define@boolkey{glossaries-extra.sty}{@glxtr}{indexcrossrefs}[true]{%
401   \if@glxtrindexcrossrefs
402   \else
403     \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
404   \fi
405 }
```

Switch off since this can increase the build time.

```
406 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

```
407 \newcommand*{\@glsxtr@autoindexcrossrefs}{\@glsxtr@indexcrossrefstrue}
```

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

```
408 \define@boolkey{glossaries-extra.sty}{@glsxtr@}{autoseeindex}[true]{%
409 }
410 \@glsxtr@autoseeindextrue
```

equations Provide a boolean option to automatically switch to the equation counter when in a numbered maths environment.

```
411 \define@boolkey{glossaries-extra.sty}{@glsxtr@}{equations}[true]{%
412 }
413 \@glsxtr@equationsfalse
```

\glsxtr@float

```
414 \let\glsxtr@float\@float
```

glsxtr@dblfloat

```
415 \let\glsxtr@dblfloat\@dblfloat
```

floats Provide a boolean option to automatically switch to the the corresponding counter when in a float.

```
416 \define@boolkey{glossaries-extra.sty}{@glsxtr@}{floats}[true]{%
417   \if@glsxtr@floats
418     \renewcommand*{\@float}[1]{\renewcommand{\glscounter}{##1}\glsxtr@float{##1}}%
419     \renewcommand*{\@dblfloat}[1]{\renewcommand{\glscounter}{##1}\glsxtr@dblfloat{##1}}%
420   \else
421     \let\@float\glsxtr@float
422     \let\@dblfloat\glsxtr@dblfloat
423   \fi
424 }
425 \@glsxtr@floatsfalse
```

iesExtraWarning Allow users to suppress warnings.

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

raWarningNoLine Allow users to suppress warnings.

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

429 \@glsxtr@declareoption{nowarn}{%
430   \let\GlossariesExtraWarning\@gobble
431   \let\GlossariesExtraWarningNoLine\@gobble
432   \glsxtr@doption{nowarn}%
433 }
```

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

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

`postdot` Shortcut for `nopostdot=false`

```
435 \@glxtr@declareoption{postdot}{%
436   \glxtr@doption{nopostdot=false}%
437   \renewcommand*{\@glxtr@defpostpunc}{%
438     \renewcommand*{\glspostdescription}{%
439       \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi}%
440   }%
441 }
```

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

```
442 \define@choicekey{glossaries-extra.sty}{nopostdot}{true,false}[true]{%
443   \glxtr@doption{nopostdot=#1}%
444   \renewcommand*{\@glxtr@defpostpunc}{%
445     \renewcommand*{\glspostdescription}{%
446       \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi}%
447   }%
448 }
```

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

```
449 \define@key{glossaries-extra.sty}{postpunc}{%
450   \glxtr@doption{nopostdot=false}%
451   \ifstrequal{#1}{dot}%
452   {%
453     \renewcommand*{\@glxtr@defpostpunc}{%
454       \renewcommand*{\glspostdescription}{.\spacefactor\sfcode'\.}%
455     }%
456   }%
457   {%
458     \ifstrequal{#1}{comma}%
459     {%
460       \renewcommand*{\@glxtr@defpostpunc}{%
461         \renewcommand*{\glspostdescription}{,}%
462       }%
463     }%
464     {%
465       \ifstrequal{#1}{none}%
466       {%
467         \glxtr@doption{nopostdot=true}%
468         \renewcommand*{\@glxtr@defpostpunc}{%
469           \renewcommand*{\glspostdescription}{}%
470         }%
471       }%
472     }%
473   }
```

```

473 \renewcommand*{\@glsxtr@defpostpunc}{%
474 \renewcommand*{\glspostdescription}{#1}%
475 }%
476 }%
477 }%
478 }%
479 }

```

glsxtrabbrvtype Glossary type for abbreviations.

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

bbreviationsdef Set by abbreviations option.

```
481 \newcommand*{\@glsxtr@abbreviationsdef}{}

```

bbreviationsdef

```

482 \newcommand*{\@glsxtr@doabbreviationsdef}{%
483 \ifpackageloaded{babel}%
484 {\providecommand{\abbreviationsname}{\acronymname}}%
485 {\providecommand{\abbreviationsname}{Abbreviations}}%
486 \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
487 \renewcommand*{\glsxtrabbrvtype}{abbreviations}%
488 \newcommand*{\printabbreviations}[1][{}]{%
489 \printglossary[type=\glsxtrabbrvtype,##1]%
490 }%
491 \disable@keys{glossaries-extra.sty}{abbreviations}%

If the acronym option hasn't been used, change \acronymtype to \glsxtrabbrvtype.
492 \ifglsacronym
493 \else
494 \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
495 \fi
496 }%

```

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

```

497 \@glsxtr@declareoption{abbreviations}{%
498 \let\@glsxtr@abbreviationsdef\@glsxtr@doabbreviationsdef
499 }

```

iationShortcuts 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).

```

500 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
501 \newcommand*{\ab}{\cgl}%
502 \newcommand*{\abp}{\cglsp}%
503 \newcommand*{\as}{\glsxtrshort}%
504 \newcommand*{\asp}{\glsxtrshortpl}%
505 \newcommand*{\al}{\glsxtrlong}%
506 \newcommand*{\alp}{\glsxtrlongpl}%
507 \newcommand*{\af}{\glsxtrfull}%

```

```

508 \newcommand*{\afp}{\glxstrfullpl}%
509 \newcommand*{\Ab}{\cGls}%
510 \newcommand*{\Abp}{\cGlspl}%
511 \newcommand*{\As}{\Glsxtrshort}%
512 \newcommand*{\Asp}{\Glsxtrshortpl}%
513 \newcommand*{\Al}{\Glsxtrlong}%
514 \newcommand*{\Alp}{\Glsxtrlongpl}%
515 \newcommand*{\Af}{\Glsxtrfull}%
516 \newcommand*{\Afp}{\Glsxtrfullpl}%
517 \newcommand*{\AB}{\cGLS}%
518 \newcommand*{\ABP}{\cGLSpl}%
519 \newcommand*{\AS}{\GLSxtrshort}%
520 \newcommand*{\ASP}{\GLSxtrshortpl}%
521 \newcommand*{\AL}{\GLSxtrlong}%
522 \newcommand*{\ALP}{\GLSxtrlongpl}%
523 \newcommand*{\AF}{\GLSxtrfull}%
524 \newcommand*{\AFP}{\GLSxtrfullpl}%

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

```

Disable this command after it's been used.

```

526 \let\GlsXtrDefineAbbreviationShortcuts\relax
527 }

```

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

```

528 \newcommand*{\GlsXtrDefineAcShortcuts}{%
529   \newcommand*{\ac}{\cgl}%
530   \newcommand*{\acp}{\cglsp}%
531   \newcommand*{\acs}{\glxtrshort}%
532   \newcommand*{\acsp}{\glxtrshortpl}%
533   \newcommand*{\acl}{\glxtrlong}%
534   \newcommand*{\aclp}{\glxtrlongpl}%
535   \newcommand*{\acf}{\glxtrfull}%
536   \newcommand*{\acfp}{\glxtrfullpl}%
537   \newcommand*{\Ac}{\cGls}%
538   \newcommand*{\Acp}{\cGLspl}%
539   \newcommand*{\Acs}{\Glsxtrshort}%
540   \newcommand*{\Acsp}{\Glsxtrshortpl}%
541   \newcommand*{\Acl}{\Glsxtrlong}%
542   \newcommand*{\Aclp}{\Glsxtrlongpl}%
543   \newcommand*{\Acf}{\Glsxtrfull}%
544   \newcommand*{\Acfp}{\Glsxtrfullpl}%
545   \newcommand*{\AC}{\cGLS}%
546   \newcommand*{\ACP}{\cGLSpl}%
547   \newcommand*{\ACS}{\GLSxtrshort}%
548   \newcommand*{\ACSP}{\GLSxtrshortpl}%
549   \newcommand*{\ACL}{\GLSxtrlong}%
550   \newcommand*{\ACLP}{\GLSxtrlongpl}%
551   \newcommand*{\ACF}{\GLSxtrfull}%

```

```

552 \newcommand*{\ACFP}{\GLSxtrfullpl}%
553 \providecommand*{\newabbr}{\newabbreviation}%

```

Disable this command after it's been used.

```

554 \let\GLSxtrDefineAcShortcuts\relax
555 }

```

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

```

556 \newcommand*{\GLSxtrDefineOtherShortcuts}{%
557 \newcommand*{\newentry}{\newglossaryentry}%
558 \ifdef\printsymbols
559 {%
560 \newcommand*{\newsym}{\glxtrnewsymbol}%
561 }{}%
562 \ifdef\printnumbers
563 {%
564 \newcommand*{\newnum}{\glxtrnewnumber}%
565 }{}%
566 \let\GLSxtrDefineOtherShortcuts\relax
567 }

```

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.

```

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

```

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

```

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

```

shortcuts Provide shortcuts option. Unlike the glossaries version, this is a choice rather than a boolean key but it also provides shortcuts=true and shortcuts=false, which are equivalent to 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).

```

570 \define@choicekey{glossaries-extra.sty}{shortcuts}%
571 [\@glxtr@shortcutsval\@glxtr@shortcutsnr]%
572 {acronyms,acro,abbreviations,abbr,other,all,true,ac,none,false}[true]{%
573 \ifcase\@glxtr@shortcutsnr\relax % acronyms
574 \renewcommand*{\@glxtr@setupshortcuts}{%
575 \glacrshortcuttrue
576 \DefineAcronymSynonyms
577 }%
578 \or % acro
579 \renewcommand*{\@glxtr@setupshortcuts}{%
580 \glacrshortcuttrue

```

```

581     \DefineAcronymSynonyms
582 }%
583 \or % abbreviations
584   \renewcommand*{\@glxtr@setupshortcuts}{%
585     \GlsXtrDefineAbbreviationShortcuts
586   }%
587 \or % abbr
588   \renewcommand*{\@glxtr@setupshortcuts}{%
589     \GlsXtrDefineAbbreviationShortcuts
590   }%
591 \or % other
592   \renewcommand*{\@glxtr@setupshortcuts}{%
593     \GlsXtrDefineOtherShortcuts
594   }%
595 \or % all
596   \renewcommand*{\@glxtr@setupshortcuts}{%
597     \glsacrshortcutstrue
598     \GlsXtrDefineAcShortcuts
599     \GlsXtrDefineAbbreviationShortcuts
600     \GlsXtrDefineOtherShortcuts
601   }%
602 \or % true
603   \renewcommand*{\@glxtr@setupshortcuts}{%
604     \glsacrshortcutstrue
605     \GlsXtrDefineAcShortcuts
606     \GlsXtrDefineAbbreviationShortcuts
607     \GlsXtrDefineOtherShortcuts
608   }%
609 \or % ac
610   \renewcommand*{\@glxtr@setupshortcuts}{%
611     \glsacrshortcutstrue
612     \GlsXtrDefineAcShortcuts
613   }%
614 \else % none, false
615   \renewcommand*{\@glxtr@setupshortcuts}{}%
616 \fi
617 }

```

Leave none and false as last option.

```

614 \else % none, false
615   \renewcommand*{\@glxtr@setupshortcuts}{}%
616 \fi
617 }

```

`\lsxtr@doaccsupp`

```

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

```

glossaries-accsupp can't be loaded after glossaries-extra. glossaries-accsupp v4.29+ checks `\@glxtr@doaccsupp` to determine if it's been loaded too late.

accsupp If accsupp, load glossaries-accsupp package.

```

619 \@glsxtr@declareoption{accsupp}{%
620 \renewcommand*{\@glsxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

```

tr@doloadprefix

```

621 \newcommand*{\@glsxtr@doloadprefix}{}

```

prefix If prefix, load glossaries-prefix package.

```

622 \@glsxtr@declareoption{prefix}{%
623 \renewcommand*{\@glsxtr@doloadprefix}{\RequirePackage{glossaries-prefix}}}

```

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

```

624 \newcommand{\glsxtrNoGlossaryWarning}[1]{%
625 \GlossariesExtraWarning{Glossary ‘#1’ is missing}%
626 \@glsxtr@defaultnoglossarywarning{#1}%
627 }

```

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

```

628 \define@choicekey{glossaries-extra.sty}{nomissingglstext}
629 [ \@glsxtr@nomissingglstextval \@glsxtr@nomissingglstextnr ]%
630 {true,false}[true]{%
631 \ifcase \@glsxtr@nomissingglstextnr \relax % true
632 \renewcommand{\glsxtrNoGlossaryWarning}[1]{\null}%
633 \else % false
634 \renewcommand{\glsxtrNoGlossaryWarning}[1]{%
635 \@glsxtr@defaultnoglossarywarning{#1}%
636 }%
637 \fi
638 }

```

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

xtr@redefstyles

```

639 \newcommand*{\@glsxtr@redefstyles}{}

```

stylemods

```

640 \define@key{glossaries-extra.sty}{stylemods}[default]{%
641 \ifstrequal{#1}{default}%
642 {%
643 \renewcommand*{\@glsxtr@redefstyles}{%
644 \RequirePackage{glossaries-extra-stylemods}}%
645 }%
646 {%
647 \ifstrequal{#1}{all}%
648 {%
649 \renewcommand*{\@glsxtr@redefstyles}{%
650 \PassOptionsToPackage{all}{glossaries-extra-stylemods}%

```



```

651     \RequirePackage{glossaries-extra-stylemods}%
652 }%
653 }%
654 {%
655     \renewcommand*{\@glxtr@redefstyles}{}%
656     \@for\@glxtr@tmp:=#1\do{%
657         \IfFileExists{glossary-\@glxtr@tmp.sty}%
658         {%
659             \eappto\@glxtr@redefstyles{%
660                 \noexpand\RequirePackage{glossary-\@glxtr@tmp}}%
661             }%
662         {%
663             \PackageError{glossaries-extra}%
664                 {Glossaries style package ‘glossary-\@glxtr@tmp.sty’
665                     doesn’t exist (did you mean to use the ‘style’ key?)}%
666                 {The list of values (#1) in the ‘stylemods’ key should
667                     match the glossary-xxx.sty files provided with
668                     glossaries.sty}%
669             }%
670         }%
671     \appto\@glxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
672 }
673 }%
674 }

```

glxtr@do@style

```

675 \newcommand*{\@glxtr@do@style}{}

```

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

```

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

```

Defer actual style change:

```

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

```

Set this as the default style:

```

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

```

Set this style:

```

679 \setglossarystyle{#1}%

```

```

680 }%

```

```

681 }

```

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.

```

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

```

cationHyperlink

$\backslash\mathrm{glxtrinternallocationhyperlink}\{\langle counter\rangle\}\{\langle prefix\rangle\}\{\langle location\rangle\}$

The first two arguments are always control sequences.

```
683 \newcommand*\GlsXtrInternalLocationHyperlink}[3]{%
684   \glxtrhyperlink{#1#2#3}{#3}%
685 }
```

locationhyperlink

```
686 \newcommand*\@glxtr@wrglossary@locationhyperlink}[3]{%
687   \pageref{wrglossary.#3}%
688 }
```

indexcounter

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

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

```
689 \@glxtr@declareoption{indexcounter}{%
690   \glxtr@doption{counter=wrglossary}%
691   \ifundef\c@wrglossary
692   {%
693     \newcounter{wrglossary}%
694     \renewcommand{\thewrglossary}{\arabic{wrglossary}}%
695   }%
696   {}%
697   \renewcommand*\@glxtr@inc@wrglossaryctr}[1]{%
```

Only increment if the current counter is wrglossary.

```
698   \ifdefstring\@gl@counter{wrglossary}%
699   {%
700     \refstepcounter{wrglossary}%
701     \label{wrglossary.\thewrglossary}%
702   }%
703   {}%
704 }%
705 \renewcommand*\GlsXtrInternalLocationHyperlink}[3]{%
706   \ifdefstring\gl@entrycounter{wrglossary}%
707   {%
708     \@glxtr@wrglossary@locationhyperlink{##1}{##2}{##3}%
709   }%
710   {\glxtrhyperlink{##1##2##3}{##3}}%
711 }%
712 }
```

glxtrwrglossmark

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

```
713 \newcommand*\@glxtrwrglossmark{}\}
```

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

```
714 \newcommand*{\@@glsxtrwrglossmark}{}
715 \AtBeginDocument{\renewcommand*{\@@glsxtrwrglossmark}{\@@glsxtrwrglossmark}}
```

sxtrwrglossmark Does nothing by default.

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

debug Provide extra debug options.

```
717 \define@choicekey{glossaries-extra.sty}{debug}
718 [ \@@glsxtr@debugval \@@glsxtr@debugnr ]%
719 {true,false,showtargets,showwrgloss,all,showaccsupp}[true]{%
720   \ifcase\@@glsxtr@debugnr\relax % true
721     \glsxtr@doption{debug=true}%
722     \renewcommand*{\@@glsxtrwrglossmark}{}%
723   \or % false
724     \glsxtr@doption{debug=false}%
725     \renewcommand*{\@@glsxtrwrglossmark}{}%
726   \or % showtargets
727     \glsxtr@doption{debug=showtargets}%
728   \or % showwrgloss
729     \glsxtr@doption{debug=true}%
730     \renewcommand*{\@@glsxtrwrglossmark}{\glsxtrwrglossmark}%
731   \or % all
732     \glsxtr@doption{debug=showtargets,debug=showaccsupp}%
733     \renewcommand*{\@@glsxtrwrglossmark}{\glsxtrwrglossmark}%
734   \or % showaccsupp
735     \glsxtr@doption{debug=showaccsupp}%
736   \fi
737 }
```

Pass all other options to glossaries.

```
738 \DeclareOptionX*{%
739   \expandafter\glsxtr@doption\expandafter{\CurrentOption}}
```

Process options.

```
740 \ProcessOptionsX
```

Load glossaries if not already loaded.

```
741 \RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
742 \@glsxtr@doaccsupp
```

Load the glossaries-prefix package if required.

```
743 \@glsxtr@doloadprefix
```

Redefine \glspostdescription if required.

```
744 \@glsxtr@defpostpunc
```

`\glsshowtarget` This command was introduced to glossaries v4.32 so it may not be defined. Therefore it's defined here using `\def`. `\glsshowtargetouter` was introduced in glossaries v4.45, so that also may not be defined.

```

745 \ifdef\glsshowtargetouter
746 {
747   \renewcommand*{\glsshowtarget}[1]{%
748     \glstrtitleorpdforheading
749     {%
750       \ifmmode
751         \nfss@text{\glsshowtargetfont [#1]}%
752       \else
753         \ifinner
754           {\glsshowtargetfont [#1]}%
755         \else
756           \glsshowtargetouter{#1}%
757         \fi
758       \fi
759     }%
760     {[#1]}%
761     {\protect\glsshowtargetfont [#1]}}%
762   }
763 }
764 {

```

Old definition.

```

765 \def\glsshowtarget#1{%
766   \glstrtitleorpdforheading
767   {%
768     \ifmmode
769       \texttt{\small [#1]}%
770     \else
771       \ifinner
772         \texttt{\small [#1]}%
773       \else
774         \marginpar{\texttt{\small #1}}%
775       \fi
776     \fi
777   }%
778   {[#1]}%
779   {\texttt{\small [#1]}}%
780   }
781 }

```

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

```

782 \let\@glstr@org@doseeglossary\@do@seeglossary

```

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

```

783 \newcommand*{\@glstr@doseeglossary}[2]{%
784   \glstoifexists{#1}%

```

```

785  {%
786    \@glsxtrwrglossmark
787    \@glsxtr@org@doseeglossary{#1}{#2}%
788  }%
789 }

oindex@glossary
790 \newcommand*{\@glsxtr@dosee@alsoindex@glossary}[2]{%
791   \@glsxtr@recordsee{#1}{#2}%
792   \@glsxtr@doseeglossary{#1}{#2}%
793 }

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

                  Check if user tried autoseeindex=false when it can't be supported.
795 \if@glsxtr@autoseeindex
796 \else
797   \ifdef\@glsxtr@org@gloautosee
798   {%
799     {\PackageError{glossaries-extra}{‘autoseeindex=false’ package
800       option requires at least v4.30 of glossaries.sty}%
801     {You need to update the glossaries.sty package}%
802   }
803 \fi

\@glo@autosee  If \@glo@autosee has been defined (glossaries v4.30 onwards), redefine it to test the au-
               toseeindex option.
804 \ifdef\@glo@autosee
805 {%
806   \renewcommand*{\@glo@autosee}{%
807     \if@glsxtr@autoseeindex\@glsxtr@org@gloautosee\fi}%
808 }%
809 {}

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.
810 \renewcommand*{\gls@checkseeallowed}{%
811   \if@glsxtr@autoseeindex\@gls@see@noindex\fi
812 }

                  Define abbreviations glossaries if required.
813 \@glsxtr@abbreviationsdef
814 \let\@glsxtr@abbreviationsdef\relax

                  Setup shortcuts if required.
815 \@glsxtr@setupshortcuts

```

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

```

816 \@glsxtr@redef@for@gl@sentries

```

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

```

817 \renewcommand{\glsxtr@doption}[1]{\setupglossaries{#1}}%

```

Disable options that can only be used when the package is loaded:

```

818 \disable@keys{glossaries-extra.sty}{accsupp}

```

Now define the user command:

```

819 \newcommand*{\glossariesextrasetup}[1]{%
820   \let\glsxtr@setup@record\relax
821   \let\glsxtr@setup@shortcuts\relax
822   \let\glsxtr@redef@for@gl@sentries\relax
823   \let\glsxtr@doload@prefix\relax
824   \setkeys{glossaries-extra.sty}{#1}%
825   \@glsxtr@abbreviationsdef
826   \let\glsxtr@abbreviationsdef\relax
827   \@glsxtr@setup@shortcuts
828   \glsxtr@setup@record
829   \@glsxtr@redef@for@gl@sentries
830   \@glsxtr@doload@prefix
831 }

```

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

```

832 \let\glsxtr@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.

```

833 \newcommand*{\glsxtr@@do@wrglossary}[1]{%
834   \@@glsxtrwrglossmark
835   \glsxtr@inc@wrglossaryctr{#1}%
836   \glsxtr@org@@do@wrglossary{#1}%
837 }

```

`aveentrycounter` Save original definition of `\@gls@saveentrycounter`.

```

838 \let\glsxtr@saveentrycounter\@gls@saveentrycounter

```

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

```

839 \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter

```

`etcounterprefix` This command is provided by the base glossaries package, but is redefined here. The standard indexing methods don't directly store the hypertarget but instead need to split it into the counter, prefix and location parts, which can be reconstituted in the location list. Unfortunately, not all targets are in this form, so the links fail. With `record=nameref`, the complete target name can be saved, so this modification adjusts the warning.

```

840 \renewcommand*\@gls@getcounterprefix[2]{%
841   \protected@edef\@gls@thisloc{#1}\protected@edef\@gls@thisHloc{#2}%
842   \ifx\@gls@thisloc\@gls@thisHloc
843     \def\@glo@counterprefix{}%
844   \else
845     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
846       \def\@glo@tmp{##2}%
847       \ifx\@glo@tmp\@empty
848         \def\@glo@counterprefix{}%
849       \else
850         \def\@glo@counterprefix{##1}%
851       \fi
852     }%
853   \@gls@get@counterprefix#2.#1\end@getprefix

```

Warn if no prefix can be formed, unless record=nameref.

```

854   \ifx\@glo@counterprefix\@empty
855     \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
856     \else
857       \GlossariesExtraWarning{Hyper target ‘#2’ can’t be formed by
858       prefixing^^Jlocation ‘#1’. You need to modify the
859       definition of \string\theH\@gls@counter^^Jotherwise you
860       will get the warning: “name{\@gls@counter.#1}’ has been^^J
861       referenced but does not exist”%
862       \ifx\@glxtr@record@setting\@glxtr@record@setting@only
863       . You may want to consider using record=nameref instead%
864     \fi}%
865   \fi
866 \fi
867 \fi
868 }

```

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

sxtrdialecthook

```

869 \newcommand*\@glxtrdialecthook{}

```

Set up record option if required.

```

870 \glxtr@setup@record

```

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

```

871 \AtBeginDocument{%
872   \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
873   \def\@glxtrundeftag{\glxtrundeftag}%
874 }

```

1.2 Extra Utilities

usedOrUndefined

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

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

```
875 \newcommand*\GlsXtrIfUnusedOrUndefined}[3]{%
876   \ifglentryexists{#1}%
877   {\ifbool{glo@glsetoklabel{#1}@flag}{#3}{#2}}%
878   {#2}%
879 }
```

ifemptyglossary

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

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

```
880 \newcommand{\glxtrifemptyglossary}[3]{%
881   \ifcsdef{glolist@#1}%
882   {%
883     \ifcsstring{glolist@#1}{,}{#2}{#3}%
884   }%
885   {%
886     \glxtrundefaction{Glossary type '#1' doesn't exist}{}%
887     #2%
888   }%
889 }
```

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

```
890 \newcommand*\glxtrifkeydefined}[3]{%
891   \key@ifundefined{glossentry}{#1}{#3}{#2}%
892 }
```

ovidestoragekey Like `\gladdstoragekey` but does nothing if the key has already been defined.

```
893 \newcommand*\glxtrprovidestoragekey{%
894   \@ifstar\sglsxtr@provide@storagekey\glxtr@provide@storagekey
895 }
```


vide@storagekey Unstarred version.

```

896 \newcommand*{\@glxtr@provide@storagekey}[3]{%
897   \key@ifundefined{glossentry}{#1}%
898   {%
899     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
900     \appto\@gls@keymap{,{#1}{#1}}%
901     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
902     \appto\@newglossaryentryposthook{%
903       \letcs{\@glo@tmp}{@glo@#1}%
904       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
905     }%

```

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

```

906   \ifblank{#3}
907   {}%
908   {%
909     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
910   }%
911 }%
912 {%

```

Provide the no-link command if not already defined.

```

913   \ifblank{#3}
914   {}%
915   {%
916     \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
917   }%
918 }%
919 }

```

vide@storagekey Starred version.

```

920 \newcommand*{\s@glxtr@provide@storagekey}[1]{%
921   \key@ifundefined{glossentry}{#1}%
922   {%
923     \expandafter\newcommand\expandafter*\expandafter
924     {\csname gls@assign@#1@field\endcsname}[2]{%
925       \@gls@expand@field{##1}{#1}{##2}%
926     }%
927   }%
928   {}%
929   \@glxtr@provide@addstoragekey{#1}%
930 }

```

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

```

\GlsXtrFmtField
931 \newcommand{\GlsXtrFmtField}{useri}

tDefaultOptions
932 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

\glstrfmt The post-link hook isn't done. This now has a starred form that checks for a final optional
argument.
933 \newrobustcmd*{\glstrfmt}{\@ifstar\s@glstrfmt\@glstrfmt}

\@glstrfmt Unstarred form.
934 \newcommand*{\@glstrfmt}[3][\@glstrfmt{#1}{#2}{#3}{}

\s@glstrfmt Starred form.
935 \newcommand*{\s@glstrfmt}[3][\@glstrfmt{#1}{#2}{#3}{}
936 \new@ifnextchar[\s@glstrfmt{#1}{#2}{#3}{}%
937 {\@glstrfmt{#1}{#2}{#3}{}%
938 }

\s@glstrfmt Pick up final optional argument.
939 \def\s@glstrfmt#1#2#3[#4]{\@glstrfmt{#1}{#2}{#3}{#4}}

\@glstrfmt Actual inner working.
940 \newcommand*{\@glstrfmt}[4]{%
    Since there's no post-link hook to worry about, grouping can be added to provide some pro-
    tection against nesting (but in general nested link text should be avoided).
941 \begingroup
942 \def\glslabel{#2}%
943 \glsoifexistsordo{#2}%
944 {%
945 \ifglshasfield{\GlsXtrFmtField}{#2}%
946 {%
947 \let\do@glslink@checkfirsthyper\relax
948 \expandafter\@glslink\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
949 {\glstrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
950 }%
951 {\glstrfmtdisplay{@firstofone}{#3}{#4}}%
952 }%
953 {%
    Has the default noindex been counteracted? If so, this needs \glsadd in case bib2gls needs
    to pick up the record.
954 \begingroup
955 \@glslsetdefault@glslink@opts
956 \setkeys{glslink}{\GlsXtrFmtDefaultOptions,#1}%
957 \ifKV@glslink@noindex\else\glsadd{#2}\fi
958 \endgroup

```

```

959     \glstrfmtdisplay{@firstofone}{#3}{#4}%
960   }%
961 \endgroup
962 }

\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).
963 \newcommand{\glstrfmtdisplay}[3]{\csuse{#1}{#2}#3}

\glxtrentryfmt No link or indexing.
964 \ifdef\texorpdfstring
965 {
966   \newcommand*{\glxtrentryfmt}[2]{%
967     \texorpdfstring{\@glxtrentryfmt{#1}{#2}}{\glstrpdfentryfmt{#1}{#2}}%
968   }
969 }
970 {
971   \newcommand*{\glxtrentryfmt}{\@glxtrentryfmt}
972 }

\glstrpdfentryfmt Use for the PDF bookmarks.
973 \newcommand*{\glstrpdfentryfmt}[2]{#2}

\@glxtrentryfmt
974 \newrobustcmd*{\@glxtrentryfmt}[2]{%
  Locally define \glslabel in case the helper command needs to access the label.
975   {%
976     \edef\glslabel{#1}%
977     \glsdoidexistsordo{#1}%
978     {%
979       \ifglshasfield{\GlsXtrFmtField}{#1}%
980       {%
981         \csuse{\glscurrentfieldvalue}{#2}%
982       }%
983       {#2}%
984     }%
985     {#2}%
986   }%
987 }

\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.
988 \newcommand*{\glstrfieldlistadd}[3]{%
989   \listcsadd{glo@\glsdetoklabel{#1}@#2}{#3}%
990 }

```

trfieldlistgadd Similarly but uses \listcsgadd.

```

991 \newcommand*{\glxtrfieldlistgadd}[3]{%
992   \listcsgadd{glo@\glsdetoklabel{#1}@\#2}@\#3}%
993 }
```

trfieldlistead Similarly but uses \listcseadd.

```

994 \newcommand*{\glxtrfieldlistead}[3]{%
995   \listcseadd{glo@\glsdetoklabel{#1}@\#2}@\#3}%
996 }
```

trfieldlistxadd Similarly but uses \listcsxadd.

```

997 \newcommand*{\glxtrfieldlistxadd}[3]{%
998   \listcsxadd{glo@\glsdetoklabel{#1}@\#2}@\#3}%
999 }
```

Now provide commands to iterate over these lists.

fieldddolistloop

```

1000 \newcommand*{\glxtrfieldddolistloop}[2]{%
1001   \dolistcsloop{glo@\glsdetoklabel{#1}@\#2}%
1002 }
```

fieldforlistloop

```

1003 \newcommand*{\glxtrfieldforlistloop}[3]{%
1004   \forlistcsloop{\#3}{glo@\glsdetoklabel{#1}@\#2}%
1005 }
```

fieldformatlist

```

1006 \newrobustcmd*{\glxtrfieldformatlist}[2]{%
1007   \begingroup
1008   \def\@dtl@formatlist@itemsep{}%
1009   \def\@dtl@formatlist@lastitem{}%
1010   \def\@dtl@formatlist@prelastitem{}%
1011   \def\@dtl@formatlist@prelastitemsep{}%
1012   \forlistcsloop{\@dtl@formatlist@handler}{glo@\glsdetoklabel{#1}@\#2}%
1013   \@dtl@formatlist@prelastitem\@dtl@formatlist@lastitem
1014   \endgroup
1015 }
```

List element tests:

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

```

1016 \newcommand*{\glxtrfieldifinlist}[5]{%
1017   \ifinlistcs{\#3}{glo@\glsdetoklabel{#1}@\#2}@\#4}@\#5}%
1018 }
```

rfielddxifinlist Expands item.

```
1019 \newcommand*{\glxtrfielddxifinlist}[5]{%
1020   \xifinlistcs{#3}{glo@\glsetoklabel{#1}@#2}{#4}{#5}%
1021 }
```

sxtrforcsvfield

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

```
1022 \newcommand*{\glxsxtrforcsvfield}[3]{%
1023   \@glxsxtrifhasfield{#2}{#1}%
1024   {%
1025     \let\glxsxtrendfor\@endfortrue
1026     \@for\@glxsxtr@label:=\glscurrentfieldvalue\do
1027       {\expandafter#3\expandafter{\@glxsxtr@label}}}%
1028   }%
1029 }
```

ldformatcsvlist

```
1030 \newrobustcmd*{\glxsxtrfieldformatcsvlist}[2]{%
1031   \@glxsxtrifhasfield{#2}{#1}%
1032   {\@dtlformatlist\glscurrentfieldvalue}%
1033   }%
1034 }
```

dValueInCsvList

`\GlsXtrIfFieldValueInCsvList{<label>}{<field>}{<list>}{<true>}{<false>}`

```
1035 \newcommand*{\GlsXtrIfFieldValueInCsvList}{%
1036   \@ifstar\s@GlsXtrIfFieldValueInCsvList\@GlsXtrIfFieldValueInCsvList
1037 }
```

Note \DTLifinlist performs one level on the list but not the element.

dValueInCsvList Unstarred version.

```
1038 \newcommand*{\@GlsXtrIfFieldValueInCsvList}[5]{%
1039   \@glxsxtrifhasfield{#2}{#1}%
1040   {%
1041     \expandafter\DTLifinlist\expandafter{\glscurrentfieldvalue}%
1042     {#3}{#4}{#5}%
1043   }%
1044   {#5}%
1045 }
```

dValueInCsvList Starred version.

```

1046 \newcommand*{\s@GlsXtrIfFieldValueInCsvList}[5]{%
1047   \s@glstrifhasfield{#2}{#1}%
1048   {%
1049     \expandafter\DTLifinlist\expandafter{\glscurrentfieldvalue}%
1050     {#3}{#4}{#5}%
1051   }%
1052   {#5}%
1053 }

```

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.

```

1054 \newrobustcmd{\glstrifhasfield}{%
1055   \@ifstar{\s@glstrifhasfield}{\@glstrifhasfield}%
1056 }

```

lsxtrifhasfield Unstarred version adds grouping.

```

1057 \newcommand{\@glstrifhasfield}[4]{%
1058   {\s@glstrifhasfield{#1}{#2}{#3}{#4}}%
1059 }

```

lsxtrifhasfield Starred version omits grouping.

```

1060 \newcommand{\s@glstrifhasfield}[4]{%
1061   \letcs{\glscurrentfieldvalue}{glo@\glsdetoklabel{#2}@#1}%
1062   \ifundef\glscurrentfieldvalue
1063     {#4}%
1064     {%
1065       \ifdefempty\glscurrentfieldvalue{#4}{#3}%
1066     }%
1067 }

```

rIfFieldNonZero Designed for numeric fields.

```

1068 \newcommand{\GlsXtrIfFieldNonZero}{%
1069   \@ifstar\s@GlsXtrIfFieldNonZero\@GlsXtrIfFieldNonZero
1070 }

```

rIfFieldNonZero

```

1071 \newcommand{\@GlsXtrIfFieldNonZero}[4]{%
1072   \@GlsXtrIfFieldCmpNum{#1}{#2}{=}{0}{#4}{#3}%
1073 }

```

XtrIfFieldEqNum

$\backslash\text{GlsXtrIfFieldEqNum}\{\langle field \rangle\}\{\langle label \rangle\}\{\langle value \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

Designed for numeric fields.

```

1074 \newcommand{\GlsXtrIfFieldEqNum}{%
1075   \ifstar\s@GlsXtrIfFieldEqNum\@GlsXtrIfFieldEqNum
1076 }

```

XtrIfFieldEqNum

```

1077 \newcommand{\@GlsXtrIfFieldEqNum}[5]{%
1078   \@GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
1079 }

```

XtrIfFieldEqNum

```

1080 \newcommand{\s@GlsXtrIfFieldEqNum}[5]{%
1081   \s@GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
1082 }

```

trIfFieldCmpNum

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

Designed for numeric fields.

```

1083 \newcommand{\GlsXtrIfFieldCmpNum}{%
1084   \ifstar\s@GlsXtrIfFieldCmpNum\@GlsXtrIfFieldCmpNum
1085 }

```

trIfFieldCmpNum

```

1086 \newcommand{\@GlsXtrIfFieldCmpNum}[6]{%
1087   {%
1088     \letcs{\glscurrentfieldvalue}{glo@glstdetoklabel{#2}@#1}%
1089     \ifundef\glscurrentfieldvalue
1090     {\def\glscurrentfieldvalue{0}}%
1091     {%
1092       \ifdefempty\glscurrentfieldvalue
1093       {\def\glscurrentfieldvalue{0}}%
1094       }%
1095     }%
1096     \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
1097   }%
1098 }

```

trIfFieldCmpNum

```

1099 \newcommand{\s@GlsXtrIfFieldCmpNum}[6]{%
1100   \letcs{\glscurrentfieldvalue}{glo@glstdetoklabel{#2}@#1}%
1101   \ifundef\glscurrentfieldvalue
1102   {\def\glscurrentfieldvalue{0}}%
1103   {%
1104     \ifdefempty\glscurrentfieldvalue
1105     {\def\glscurrentfieldvalue{0}}%

```

```

1106   {}%
1107   }%
1108   \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
1109 }

```

XtrIfFieldUndef

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

Just uses \ifcsundef.

```

1110 \newcommand{\GlsXtrIfFieldUndef}[2]{%
1111   \ifcsundef{glo@\glsdetoklabel{#2}@#1}%
1112 }

```

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

```

1113 \newcommand*{\glsxtrusefield}[2]{%
1114   \@gls@entry@field{#1}{#2}%
1115 }

```

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

```

1116 \ifdef\teorpdfstring
1117 {
1118   \newcommand*{\Glsxtrusefield}[2]{%
1119     \teorpdfstring
1120       {\@Gls@entry@field{#1}{#2}}
1121       {\@gls@entry@field{#1}{#2}}%
1122   }
1123 }
1124 {
1125   \newcommand*{\Glsxtrusefield}[2]{%
1126     \@Gls@entry@field{#1}{#2}%
1127   }
1128 }

```

\GLSxtrusefield As above but convert to all caps.

```

1129 \ifdef\teorpdfstring
1130 {
1131   \newcommand*{\GLSxtrusefield}[2]{%
1132     \teorpdfstring
1133       {\glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}}%
1134       {\@gls@entry@field{#1}{#2}}%
1135   }
1136 }
1137 {
1138   \newcommand*{\GLSxtrusefield}[2]{%
1139     \glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}%

```



```

1140 }
1141 }

entryparentname
1142 \newcommand*{\glxtrentryparentname}[1]{%
1143   \ifcsdef{glo@\glstdetoklabel{#1}@parent}%
1144     {\csuse{glo@\csuse{glo@\glstdetoklabel{#1}@parent}@name}}%
1145     {}%
1146 }

\glxtrdeffield Just use \csdef to provide a field value for the given entry.
1147 \newcommand*{\glxtrdeffield}[2]{\csdef{glo@\glstdetoklabel{#1}@#2}}

glxtrdeffield Just use \csedef to provide a field value for the given entry.
1148 \newcommand*{\glxtrdeffield}[2]{\protected@csedef{glo@\glstdetoklabel{#1}@#2}}

etfieldifexists
1149 \newcommand*{\glxtrsetfieldifexists}[3]{\glstdoifexists{#1}{#3}}

\GlsXtrSetField Allow the user to set a field. First argument entry label, second argument field label, third
argument value.
1150 \newrobustcmd*{\GlsXtrSetField}[3]{%
1151   \glxtrsetfieldifexists{#1}{#2}%
1152   {\csdef{glo@\glstdetoklabel{#1}@#2}{#3}}%
1153 }

\GlsXtrLetField Uses \cslet instead. Third argument should be a macro.
1154 \newrobustcmd*{\GlsXtrLetField}[3]{%
1155   \glxtrsetfieldifexists{#1}{#2}%
1156   {\cslet{glo@\glstdetoklabel{#1}@#2}{#3}}%
1157 }

sGlsXtrLetField Uses \csletcs instead. Third argument should be a control sequence name.
1158 \newrobustcmd*{\csGlsXtrLetField}[3]{%
1159   \glxtrsetfieldifexists{#1}{#2}%
1160   {\csletcs{glo@\glstdetoklabel{#1}@#2}{#3}}%
1161 }

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.
1162 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
1163   \glxtrsetfieldifexists{#1}{#2}%
1164   {\csletcs{glo@\glstdetoklabel{#1}@#2}{glo@\glstdetoklabel{#3}@#4}}%
1165 }

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

```

```

1166 \newrobustcmd*{\glsXtrSetField}[3]{%
1167   \glstrsetfieldifexists{#1}{#2}%
1168   {\csgdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
1169 }

```

xGlsXtrSetField

```

1170 \newrobustcmd*{\xGlsXtrSetField}[3]{%
1171   \glstrsetfieldifexists{#1}{#2}%
1172   {\protected@csxdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
1173 }

```

eGlsXtrSetField

```

1174 \newrobustcmd*{\eGlsXtrSetField}[3]{%
1175   \glstrsetfieldifexists{#1}{#2}%
1176   {\protected@csedef{glo@\glsdetoklabel{#1}@#2}{#3}}%
1177 }

```

XtrIfFieldEqStr Starred version uses starred version of \glstrifhasfield (that is, no grouping).

```

1178 \newcommand*{\GlsXtrIfFieldEqStr}{%
1179   \@ifstar\s@GlsXtrIfFieldEqStr\@GlsXtrIfFieldEqStr
1180 }

```

XtrIfFieldEqStr

```

1181 \newrobustcmd*{\@GlsXtrIfFieldEqStr}[5]{%
1182   \@glstrifhasfield{#1}{#2}%
1183   {%
1184     \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
1185   }%
1186   {#5}%
1187 }

```

XtrIfFieldEqStr

```

1188 \newrobustcmd*{\s@GlsXtrIfFieldEqStr}[5]{%
1189   \s@glstrifhasfield{#1}{#2}%
1190   {%
1191     \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
1192   }%
1193   {#5}%
1194 }

```

rIfFieldEqXpStr Like the above but first expands the string. Starred version uses starred version of \glstrifhasfield (that is, no grouping).

```

1195 \newcommand*{\GlsXtrIfFieldEqXpStr}{%
1196   \@ifstar\s@GlsXtrIfFieldEqXpStr\@GlsXtrIfFieldEqXpStr
1197 }

```

rIfFieldEqXpStr

```

1198 \newrobustcmd*{\@GlsXtrIfFieldEqXpStr}[5]{%

```

```

1199 \@glxtrifhasfield{#1}{#2}%
1200 {%
1201   \protected@edef\@gls@tmp{#3}%
1202   \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
1203 }%
1204 {#5}%
1205 }

```

rIfFieldEqXpStr

```

1206 \newrobustcmd*{\s@GlsXtrIfFieldEqXpStr}[5]{%
1207   \s@glxtrifhasfield{#1}{#2}%
1208   {%
1209     \protected@edef\@gls@tmp{#3}%
1210     \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
1211   }%
1212   {#5}%
1213 }

```

fXpFieldEqXpStr Like the above but also expands the field value. Starred version uses starred version of \glxtrifhasfield (that is, no grouping).

```

1214 \newcommand*{\GlsXtrIfXpFieldEqXpStr}{%
1215   \@ifstar\s@GlsXtrIfXpFieldEqXpStr\@GlsXtrIfXpFieldEqXpStr
1216 }

```

fXpFieldEqXpStr

```

1217 \newrobustcmd*{\@GlsXtrIfXpFieldEqXpStr}[5]{%
1218   \@glxtrifhasfield{#1}{#2}%
1219   {%
1220     \protected@edef\@gls@tmp{\glscurrentfieldvalue}%
1221     \let\glscurrentfieldvalue\@gls@tmp
1222     \protected@edef\@gls@tmp{#3}%
1223     \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
1224   }%
1225   {#5}%
1226 }

```

fXpFieldEqXpStr

```

1227 \newrobustcmd*{\s@GlsXtrIfXpFieldEqXpStr}[5]{%
1228   \s@glxtrifhasfield{#1}{#2}%
1229   {%
1230     \protected@edef\@gls@tmp{\glscurrentfieldvalue}%
1231     \let\glscurrentfieldvalue\@gls@tmp
1232     \protected@edef\@gls@tmp{#3}%
1233     \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
1234   }%
1235   {#5}%
1236 }

```

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

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

```
1237 \ifdef\foreignlanguage
1238 {
1239   \ifdef\GetTrackedDialectFromLanguageTag
1240   {
1241     \newcommand{\GlsXtrForeignText}[2]{%
```

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

```
1242     \let\@glsxtr@org@currentfieldvalue\glscurrentfieldvalue
1243     \glsxtrifhasfield{\GlsXtrForeignTextField}{#1}%
1244     {%
1245       \expandafter\GetTrackedDialectFromLanguageTag\expandafter
1246       {\glscurrentfieldvalue}{\@glsxtr@dialect}%
1247       \let\@glsxtr@locale\glscurrentfieldvalue
1248       \let\glscurrentfieldvalue\@glsxtr@org@currentfieldvalue
1249       \ifdefempty\@glsxtr@dialect
1250       {%
```

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

```
1251     \ifundef\TrackedDialectClosestSubMatch
1252     {%
1253       \GlossariesExtraWarning{Can't obtain dialect label
1254       (tracklang v1.3.6+ required)}}%
1255     }%
1256     {\let\@glsxtr@dialect\TrackedDialectClosestSubMatch}%
1257   }%
1258   }%
1259   \ifdefempty\@glsxtr@dialect
1260   {%
```

No tracked dialect found for the root language.

```
1261   }%
1262   {%
```

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

```
1263     \ifcsundef{captions\@glsxtr@dialect}{}%
1264     {%
```

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

```
1265     \IfTrackedDialectHasMapping{\@glsxtr@dialect}%
1266     {%
1267       \edef\@glsxtr@dialect{%
1268         \GetTrackedDialectToMapping{\@glsxtr@dialect}}%
```

Does a caption hook exist for this?

```
1269         \ifcsundef{captions\@glsxtr@dialect}{}%
1270         {%
```

No mapping. Try root language label instead.

```
1271         \ifcsundef{captions\@tracklang@lang}{}%
1272         {%
1273         \let\@glsxtr@dialect\@tracklang@lang
1274         }%
1275         }%
1276         }%
1277         {%
```

No mapping. Try root language label instead.

```
1278         \ifcsundef{captions\@tracklang@lang}{}%
1279         {%
1280         \let\@glsxtr@dialect\@tracklang@lang
1281         }%
1282         }%
1283         }%
1284         }%
1285         \ifdefempty\@glsxtr@dialect
1286         {%
1287         \GlsXtrUnknownDialectWarning{\@glsxtr@locale}{\@tracklang@lang}%
1288         #2%
1289         }%
1290         {\foreignlanguage{\@glsxtr@dialect}{#2}}%
1291         }%
1292         {#2}% key not set
1293     }
1294 }
1295 {
1296     \newcommand{\GlsXtrForeignText}[2]{%
1297         \GlossariesExtraWarning{Can't encapsulate foreign text:
1298             tracklang v1.3.6+ required}%
1299         #2%
1300     }
1301 }
1302 }
1303 {
1304     \foreignlanguage isn't defined so just do <text>.
1305     \newcommand{\GlsXtrForeignText}[2]{#2}
1306 }
```

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

```
1306 \newcommand*{\GlsXtrForeignTextField}{userii}
```

nDialectWarning

```
1307 \newcommand*{\GlsXtrUnknownDialectWarning}[2]{%
```

```

1308 \GlossariesExtraWarning{Can't determine valid dialect label
1309   for locale '#1' (root language: #2)}%
1310 }

```

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

```

1311 \ifdef\GlsEntryCounterLabelPrefix
1312 {%
1313   \newcommand*\glstrpageref}[1]{%
1314     \ifglentrycounter
1315       \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1316     \else
1317       \ifglssubentrycounter
1318         \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
1319       \else
1320         \gls{#1}%
1321       \fi
1322     \fi
1323   }
1324 }%
1325 {%
1326   \newcommand*\glstrpageref}[1]{%
1327     \ifglentrycounter
1328       \pageref{glsentry-\glsdetoklabel{#1}}%
1329     \else
1330       \ifglssubentrycounter
1331         \pageref{glsentry-\glsdetoklabel{#1}}%
1332       \else
1333         \gls{#1}%
1334       \fi
1335     \fi
1336   }
1337 }%

```

`lossarypreamble`

```

1338 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
1339   \ifcsdef{glolist@#1}%
1340   {%
1341     \ifcsundef{@glossarypreamble@#1}%
1342     {\csdef{@glossarypreamble@#1}{}}%
1343     {}%
1344     \csappto{@glossarypreamble@#1}{#2}%
1345   }%
1346   {%
1347     \GlossariesExtraWarning{Glossary '#1' is not defined}%
1348   }%
1349 }

```

lossarypreamble

```
1350 \newcommand{\preglossarypreamble}[2][\glsdefaultttype]{%
1351   \ifcsdef{glolist@#1}%
1352   {%
1353     \ifcsundef{@glossarypreamble@#1}%
1354     {\csdef{@glossarypreamble@#1}{}}%
1355     {}%
1356     \cspretoto{@glossarypreamble@#1}{#2}%
1357   }%
1358   {%
1359     \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
1360   }%
1361 }
```

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.

The original `\@gls@entry@field` causes a problem for undefined entries when used in section headings or captions. Since entries must be defined with just the base package this isn't a significant issue, but it will cause a problem with **bib2gls** where no entries are defined on the first \LaTeX call, so redefine `\@gls@entry@field` to use `\csuse` instead of `\csname`.

`\@gls@entry@field`

`\@gls@entry@field{\langle label \rangle}{\langle field \rangle}`

This command was introduced to glossaries version 4.03 but older versions are likely to be incompatible with glossaries-extra.

```
1362 \ifdef\@gls@entry@field
1363 {
1364   \renewcommand*{\@gls@entry@field}[2]{\csuse{glo@\glsdetoklabel{#1}@#2}}
1365 }
1366 {}
```

`\ifglsused`

`\ifglsused{\langle label \rangle}{\langle true part \rangle}{\langle false part \rangle}`

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

```
1367 \renewcommand*{\ifglsused}[3]{%
```

```

1368 \glsdoifexists{#1}{\ifbool{glo@glstetoklabel{#1}@flag}{#2}{#3}}%
1369 }

```

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`

```

1370 \renewcommand*{\longnewglossaryentry}{%
1371 \ifstar\glstr@s@longnewglossaryentry\glstr@longnewglossaryentry
1372 }

```

`ewglossaryentry` Starred version.

```

1373 \newcommand{\@glstr@s@longnewglossaryentry}[3]{%
1374 \glsdoifnoexists{#1}%
1375 {%
1376 \bgroup
1377 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1378 \long\def\@newglossaryentryprehook{%
1379 \long\def\@glo@desc{#3}%
1380 \@org@newglossaryentryprehook
1381 }%
1382 \renewcommand*{\gls@assign@desc}[1]{%
1383 \global\cslet{glo@glstetoklabel{#1}@desc}{\@glo@desc}%
1384 \global\cslet{glo@glstetoklabel{#1}@descplural}{\@glo@descplural}%
1385 }
1386 \gls@defglossaryentry{#1}{#2}%
1387 \egroup
1388 }%
1389 }

```

`ewglossaryentry` Unstarred version.

```

1390 \newcommand{\@glstr@longnewglossaryentry}[3]{%
1391 \glsdoifnoexists{#1}%
1392 {%
1393 \bgroup
1394 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
1395 \long\def\@newglossaryentryprehook{%
1396 \long\def\@glo@desc{#3\glstrpostlongdescription}%
1397 \@org@newglossaryentryprehook
1398 }%
1399 \renewcommand*{\gls@assign@desc}[1]{%
1400 \global\cslet{glo@glstetoklabel{#1}@desc}{\@glo@desc}%

```

The following is different from the base `glossaries.sty`:

```

1401 \global\cslet{glo@glstetoklabel{#1}@descplural}{\@glo@descplural}%
1402 }
1403 \gls@defglossaryentry{#1}{#2}%
1404 \egroup

```



```

1405 }%
1406 }

```

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

```

1407 \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.

```

1408 \renewcommand{\newignoredglossary}{%
1409 \ifstar\glxtr@s@newignoredglossary\glxtr@org@newignoredglossary
1410 }

```

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

```

1411 \newcommand*{\glxtr@org@newignoredglossary}[1]{%
1412 \ifcsdef{glolist@#1}
1413 {%
1414 \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
1415 }%
1416 {%
1417 \ifdefempty\@ignored@glossaries
1418 {%
1419 \edef\@ignored@glossaries{#1}%
1420 }%
1421 {%
1422 \eappto\@ignored@glossaries{,#1}%
1423 }%
1424 \csgdef{glolist@#1}{,}%
1425 \ifcsundef{gls@#1@entryfmt}%
1426 {%
1427 \defglsentryfmt[#1]{\glsentryfmt}%
1428 }%
1429 {}%
1430 \ifdefempty\@gls@nohyperlist
1431 {%
1432 \renewcommand*{\@gls@nohyperlist}{#1}%
1433 }%
1434 {%
1435 \eappto\@gls@nohyperlist{,#1}%
1436 }%
1437 }%
1438 }

```

`\ignoredglossary` Starred form.

```

1439 \newcommand*{\glxtr@s@newignoredglossary}[1]{%
1440 \ifcsdef{glolist@#1}
1441 {%
1442 \glxtrundefaction{Glossary type ‘#1’ already exists}{}%

```

```

1443 }%
1444 {%
1445   \ifdefempty\@ignored@glossaries
1446   {%
1447     \edef\@ignored@glossaries{#1}%
1448   }%
1449   {%
1450     \eappto\@ignored@glossaries{,#1}%
1451   }%
1452   \csgdef{glolist@#1}{,}%
1453   \ifcsundef{gls@#1@entryfmt}%
1454   {%
1455     \defglsentryfmt[#1]{\glsentryfmt}%
1456   }%
1457   {}%
1458 }%
1459 }

```

`\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.

```

1460 \glsifusetranslator
1461 {%
1462   \renewcommand*{\glsettoctitle}[1]{%
1463     \ifcsdef{gls@tr@set@#1@toctitle}%
1464     {%
1465       \csuse{gls@tr@set@#1@toctitle}%
1466     }%
1467     {%
1468       \ifcsdef{@glotype@#1@title}%
1469       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1470       {\def\glossarytoctitle{\glossarytitle}}%
1471     }%
1472   }%
1473 }
1474 {
1475   \renewcommand*{\glsettoctitle}[1]{%
1476     \ifcsdef{@glotype@#1@title}%
1477     {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
1478     {\def\glossarytoctitle{\glossarytitle}}%
1479   }
1480 }

```

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

```

1481 \newcommand{\provideignoredglossary}{%
1482   \@ifstar\glxtr@s@provideignoredglossary\glxtr@provideignoredglossary
1483 }

```

`ignoredglossary` Unstarred version.

```

1484 \newcommand*{\glxtr@provideignoredglossary}[1]{%

```

```

1485 \ifcsdef{glolist@#1}
1486 {}%
1487 {%
1488   \ifdefempty\@ignored@glossaries
1489   {%
1490     \edef\@ignored@glossaries{#1}%
1491   }%
1492   {%
1493     \eappto\@ignored@glossaries{, #1}%
1494   }%
1495   \csgdef{glolist@#1}{,}%
1496   \ifcsundef{gls@#1@entryfmt}%
1497   {%
1498     \defglsentryfmt[#1]{\glsentryfmt}%
1499   }%
1500   {}%
1501   \ifdefempty\@gls@nohyperlist
1502   {%
1503     \renewcommand*{\@gls@nohyperlist}{#1}%
1504   }%
1505   {%
1506     \eappto\@gls@nohyperlist{, #1}%
1507   }%
1508 }%
1509 }

```

`ignoredglossary` Starred form.

```

1510 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
1511   \ifcsdef{glolist@#1}
1512   {}%
1513   {%
1514     \ifdefempty\@ignored@glossaries
1515     {%
1516       \edef\@ignored@glossaries{#1}%
1517     }%
1518     {%
1519       \eappto\@ignored@glossaries{, #1}%
1520     }%
1521     \csgdef{glolist@#1}{,}%
1522     \ifcsundef{gls@#1@entryfmt}%
1523     {%
1524       \defglsentryfmt[#1]{\glsentryfmt}%
1525     }%
1526     {}%
1527   }%
1528 }

```

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

```

1529 \newcommand*{\glstrcopytoglossary}[2]{%
1530   \glstoifexists{#1}%
1531   {%
1532     \ifcsdef{glolist@#2}
1533     {%
1534       \cseappto{glolist@#2}{#1,}%
1535     }%
1536     {%
1537       \glstrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
1538     }%
1539   }%
1540 }

```

1.3.1 Existence Checks

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

```

1541 \renewcommand{\glstoifexists}[2]{%
1542   \ifglstryexists{#1}{#2}%
1543   {%

```

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

```

1544   \edef\glslabel{\glstetoklabel{#1}}%
1545   \glstrundefaction{Glossary entry ‘\glslabel’
1546     has not been defined}{You need to define a glossary entry before
1547     you can reference it.}%
1548   }%
1549 }

```

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

```

1550 \renewcommand{\glstoifnoexists}[2]{%
1551   \ifglstryexists{#1}{%
1552     \glstrundefaction{Glossary entry ‘\glstetoklabel{#1}’
1553       has already been defined}{}}{#2}%
1554 }

```

`\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.

```

1555 \ifdef\glstoifexistsordo
1556 {%
1557   \renewcommand{\glstoifexistsordo}[3]{%
1558     \ifglstryexists{#1}{#2}%
1559     {%
1560       \glstrundefaction{Glossary entry ‘\glstetoklabel{#1}’
1561         has not been defined}{You need to define a glossary entry
1562         before you can use it.}%
1563       #3%
1564     }%
1565   }%

```

```

1566 }
1567 {%
1568   \glstr@warnonexistsordo\glstoifexistsordo
1569   \newcommand{\glstoifexistsordo}[3]{%
1570     \ifglstryexists{#1}{#2}%
1571     {%
1572       \glstrundefaction{Glossary entry ‘\glstetoklabel{#1}’
1573       has not been defined}{You need to define a glossary entry
1574       before you can use it.}%
1575       #3%
1576     }%
1577   }%
1578 }

```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```

1579 \ifdef\doifglossarynoexistsordo
1580 {%
1581   \renewcommand{\doifglossarynoexistsordo}[3]{%
1582     \ifglossaryexists{#1}%
1583     {%
1584       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1585       #3%
1586     }%
1587     {#2}%
1588   }%
1589 }
1590 {%
1591   \glstr@warnonexistsordo\doifglossarynoexistsordo
1592   \newcommand{\doifglossarynoexistsordo}[3]{%
1593     \ifglossaryexists{#1}%
1594     {%
1595       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
1596       #3%
1597     }%
1598     {#2}%
1599   }%
1600 }
1601

```

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.

```

1602 \appto\newglossaryentryposthook{%
1603   \ifdefvoid\@glo@see
1604   {\csxdef{glo@\@glo@label @see}{}}%
1605   {%
1606     \csxdef{glo@\@glo@label @see}{\@glo@see}%

```

```

1607 \if@glxtr@autoseeindex
1608 \@glxtr@autoindexcrossrefs
1609 \fi
1610 }%
1611 }
1612 \appto\@gls@keymap{,{see}{see}}

```

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

```

1613 \newcommand*{\glxtrusesee}[1]{%
1614 \glsdoifexists{#1}%
1615 {%
1616 \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
1617 \ifdefempty\@glo@see
1618 {}%
1619 {%
1620 \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
1621 }%
1622 }%
1623 }

```

`\glxtr@usesee`

```

1624 \newcommand*{\glxtr@usesee}[1][\seename]{%
1625 \@glxtr@usesee[1]%
1626 }

```

`\@glxtr@usesee`

```

1627 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
1628 \glxtruseseeformat{#1}{#2}%
1629 }

```

`\glxtruseseeformat` 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.

```

1630 \newcommand*{\glxtruseseeformat}[2]{%
1631 \glsseeformat{#1}{#2}{}%
1632 }

```

`\glsseeitemformat` 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 sanitized. Since this is no longer a problem, glossaries-extra restored the original definition as it makes more sense to use the name in the cross-reference list. Unfortunately this doesn't take style changes into account, so as from v1.42, this now uses `\glsfmttext` and `\glsfmtname` instead. (The text field is chosen rather than the short field to allow for the “noshort” styles.)

```

1633 \renewcommand*{\glsseeitemformat}[1]{%
1634 \ifglshashshort{#1}{\glsfmttext{#1}}{\glsfmtname{#1}}%
1635 }

```

`\glxtrhiername`

`\glxtrhiername{<label>}`

Displays the hierarchical name for the given entry. The cross-reference format `\glseeitemformat` may be redefined to use this command to show the hierarchy, if required. This now uses `\glsfmtext` and `\glsfmtname` instead of `\glaccessshort` and `\glaccessname` to allow for style formatting.

```
1636 \newcommand*{\glxtrhiername}[1]{%
1637   \glsoifexists{#1}%
1638   {%
1639     \glxtrifhasfield{parent}{#1}%
1640     {\glxtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep}%
1641     {}%
1642     \ifglshasshort{#1}{\glsfmtext{#1}}{\glsfmtname{#1}}%
1643   }%
1644 }
```

`\Glsxtrhiername`

`\Glsxtrhiername{<label>}`

As above but displays the top-level name with an initial capital.

```
1645 \newcommand*{\Glsxtrhiername}[1]{%
1646   \glsoifexists{#1}%
1647   {%
1648     \glxtrifhasfield{parent}{#1}%
1649     {%
1650       \Glsxtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep
1651       \ifglshasshort{#1}{\glsfmtext{#1}}{\glsfmtname{#1}}%
1652     }%
1653     {\ifglshasshort{#1}{\Glsfmtext{#1}}{\Glsfmtname{#1}}}%
1654   }%
1655 }
```

`\GlsXtrhiername`

`\GlsXtrhiername{<label>}`

As above but converts the first letter of each name to a capital. (Note that this isn't applying title case, just capitalising the start of each hierarchical element.)

```
1656 \newcommand*{\GlsXtrhiername}[1]{%
1657   \glsoifexists{#1}%
1658   {%
1659     \glxtrifhasfield{parent}{#1}%
1660     {\GlsXtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep}%
1661     {}%
1662   }%
1663 }
```

```

1662 \ifglshasshort{#1}{\Glsfmttext{#1}}{\Glsfmtname{#1}}%
1663 }%
1664 }

```

`\GLSxtrhiername`

`\GLSxtrhiername{<label>}`

As above but displays the top-level name in all-caps.

```

1665 \newcommand*{\GLSxtrhiername}[1]{%
1666 \glsdoifexists{#1}%
1667 {%
1668 \glxtrifhasfield{parent}{#1}%
1669 {%
1670 \GLSxtrhiername{\glscurrentfieldvalue}\glxtrhiernamesep
1671 \ifglshasshort{#1}{\Glsfmttext{#1}}{\Glsfmtname{#1}}%
1672 }%
1673 {\ifglshasshort{#1}{\GLSfmttext{#1}}{\GLSfmtname{#1}}}%
1674 }%
1675 }

```

`\GLSXTRhiername`

`\GLSXTRhiername{<label>}`

As above but displays all names in all-caps.

```

1676 \newcommand*{\GLSXTRhiername}[1]{%
1677 \glsdoifexists{#1}%
1678 {%
1679 \glxtrifhasfield{parent}{#1}%
1680 {\GLSXTRhiername{\glscurrentfieldvalue}\glxtrhiernamesep}%
1681 {}
1682 \ifglshasshort{#1}{\GLSfmttext{#1}}{\GLSfmtname{#1}}%
1683 }%
1684 }

```

`\glxtrhiernamesep` Separator used in `\glxtrhiername` and variants.

```

1685 \newcommand*{\glxtrhiernamesep}{\,\{\small$\triangleright$}\,}

```

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

```

1686 \newcommand*{\glxtruseseealso}[1]{%
1687 \glsdoifexists{#1}%
1688 {%
1689 \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%
1690 \ifdefempty\@glo@see

```



```

1691     {}%
1692     {%
1693         \expandafter\glxtruseseealsoformat\expandafter{\@glo@see}%
1694     }%
1695 }%
1696 }

```

`\glxtrusealias` Apply `\glssseeformat` to the alias key if not empty. There's no optional tag to worry about here. The value also isn't a comma-separated list, but use the same interface.

```

1697 \newcommand*{\glxtrusealias}[1]{%
1698     \glsdodefexists{#1}%
1699     {%
1700         \letcs{\@glo@see}{glo\glsdetoklabel{#1}@alias}%
1701         \ifdefempty{\@glo@see
1702             {}%
1703             {%

```

Expansion isn't necessary because the value is a single label not a list.

```

1704         \glxtruseseeformat{\seename}{\@glo@see}%
1705     }%
1706 }%
1707 }

```

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

```

1708 \newcommand*{\glxtrusealsoformat}[1]{%
1709     \glssseeformat[\seealso]{#1}%
1710 }

```

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

```

1711 \newrobustcmd{\glxtrseelist}[1]{%
1712     \edef\@glo@tmp{\noexpand\glssseelist{#1}}\@glo@tmp
1713 }

```

`\seealso` In case this command hasn't been defined. Languages packages actually provide `\also` so use that if it's defined.

```

1714 \ifdef\also
1715 {\providecommand{\seealso}{\also}}
1716 {\providecommand{\seealso}{see also}}

```

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

```

1717 \ifdef\@xdycrossrefhook
1718 {

```

Add the cross-reference class definition to the hook.

```

1719 \appto\@xdycrossrefhook{%
1720   \write\glswrite{(define-crossref-class \string"seealso\string"
1721     :unverified )}%
1722   \write\glswrite{(markup-crossref-list
1723     :class \string"seealso\string"^^J\space\space\space
1724     :open \string"\string\glstruseealsoformat\glsopenbrace\string"
1725     :close \string"\glsclosebrace\string")}%
1726 }

```

Append to class list.

```

1727 \appto\@xdylocationclassorder{\space\string"seealso\string"}

```

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

```

1728 \newrobustcmd*{\glstrindexseealso}[2]{%
1729   \ifx\@glstr@record@setting\@glstr@record@setting@alsoindex
1730     \@glstr@recordsee{#1}{#2}%
1731   \fi
1732   \glsdoifexists{#1}%
1733   {%
1734     \@@glstrwrglossmark
1735     \def\@gls@xref{#2}%
1736     \@onelevel@sanitize\@gls@xref
1737     \@gls@checkmkidxchars\@gls@xref
1738     \gls@glossary{\csname glo@#1@type\endcsname}{%
1739       (indexentry
1740         :tkey (\csname glo@#1@index\endcsname)
1741         :xref (\string"\@gls@xref\string")
1742         :attr \string"seealso\string"
1743       )
1744     }%
1745   }%
1746 }
1747 }
1748 {

```

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

```

1749 \newrobustcmd*{\glstrindexseealso}{\glssee[\seealsoname]}
1750 }

```

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`.

```

1751 \ifdef\gls@set@xr@key
1752 {

```

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

```

1753 \define@key{glossentry}{alias}{%
1754   \gls@set@xr@key{alias}{\@glo@alias}{#1}%
1755 }
1756 \define@key{glossentry}{seealso}{%
1757   \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
1758 }

```

Add to the key mappings.

```

1759 \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}

```

Set the default value.

```

1760 \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%

```

Assign the field values.

```

1761 \appto\@newglossaryentryposthook{%
1762   \ifdefvoid\@glo@seealso
1763     {\csxdef{glo@\@glo@label @seealso}{}}%
1764     {%
1765       \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
1766       \ifglxtr@autoseeindex
1767         \@glxtr@autoindexcrossrefs
1768       \fi
1769     }%

```

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

```

1770 \ifdefvoid\@glo@alias
1771   {\csxdef{glo@\@glo@label @alias}{}}%
1772   {%
1773     \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
1774   }%
1775 }

```

Provide user-level commands to access the values.

`\glxtralias`

```

1776 \newcommand*\glxtralias[1]{\@gls@entry@field{#1}{alias}}

```

`trseealsolabels`

```

1777 \newcommand*\glxtrseealsolabels[1]{\@gls@entry@field{#1}{seealso}}

```

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

```

1778 \appto\@glo@autoseehook{%
1779   \ifdefvoid\@glo@alias
1780     {%
1781       \ifdefvoid\@glo@seealso
1782         {}%
1783       {%
1784         \edef\@do@glsee{\noexpand\glxtrindexseealso

```

```

1785      {\@glo@label}{\@glo@seealso}}%
1786      \@do@glssee
1787    }%
1788  }%
1789  {%

```

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

```

1790      \ifdefvoid\@glo@see
1791      {%
1792        \edef\@do@glssee{\noexpand\glssee{\@glo@label}{\@glo@alias}}%
1793        \@do@glssee
1794      }%
1795    }%
1796  }%
1797 }%
1798 }
1799 {

```

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

\glsxtralias

```

1800 \glsaddstoragekey*{alias}{\glsxtralias}

```

trseealsolabels

```

1801 \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.

```

1802 \appto\@newglossaryentryposthook{%
1803   \ifcsvoid{glo@\@glo@label @alias}%
1804   {%
1805     \ifcsvoid{glo@\@glo@label @seealso}%
1806     }%
1807   {%
1808     \edef\@do@glssee{\noexpand\glsxtrindexseealso
1809       {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1810     \@do@glssee
1811   }%
1812 }%
1813 {%

```

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

```

1814   \ifdefvoid\@glo@see
1815   {%
1816     \edef\@do@glssee{\noexpand\glssee
1817       {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1818     \@do@glssee
1819   }%

```

```

1820     {}%
1821   }%
1822 }

```

```

1823 }

```

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

```

1824 \AtEndDocument{\if@glxtrindexcrossrefs\glxtraddallcrossrefs\fi}

```

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

```

1825 \newcommand*\glxtraddallcrossrefs{%
1826   \forallglossaries{\@glo@type}%
1827   {%
1828     \forallglsentries[\@glo@type]{\@glo@label}%
1829     {%
1830       \ifglsused{\@glo@label}%
1831       {\expandafter\glxtr@addunusedxrefs\expandafter{\@glo@label}}}%
1832     }%
1833   }%
1834 }

```

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

```

1835 \newcommand*\@glxtr@addunusedxrefs[1]{%
1836   \letcs{\@glo@see}{glo\glstoklabel{#1}@see}%
1837   \ifdefvoid\@glo@see
1838   {}%
1839   {%
1840     \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1841   }%
1842   \letcs{\@glo@see}{glo\glstoklabel{#1}@seealso}%
1843   \ifdefvoid\@glo@see
1844   {}%
1845   {%
1846     \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1847   }%
1848 }

```

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

```

1849 \newcommand*\glxtr@addunused[1][ ]{%
1850   \@glxtr@addunused
1851 }

```

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

```

1852 \def\@glxtr@addunused#1\@end@glxtr@addunused{%
1853   \@for\@glxtr@label:=#1\do
1854   {%
1855     \ifglsused{\@glxtr@label}}}%

```

```

1856   {%
1857     \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
1858     \glsunset{\@glsxtr@label}%
1859     \expandafter\@glsxtr@addunusedxrefs\expandafter{\@glsxtr@label}%
1860   }%
1861 }%
1862 }

```

xtrunusedformat

```

1863 \newcommand*{\glsxtrunusedformat}[1]{\unskip}

```

1.3.2 Document Definitions

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

```

1864 \ifdef\gls@begindocdefs
1865 {%
1866   \renewcommand*{\gls@begindocdefs}{%
1867     \ifnum\@glsxtr@docdefval=1\relax
1868       \@gls@enablesavenonumberlist
1869       \edef\@gls@restoreat{%
1870         \noexpand\catcode'\noexpand\@=\number\catcode'\@ \relax}%
1871       \makeatletter
1872       \InputIfFileExists{\jobname.glsdefs}{\relax}%
1873       \@gls@restoreat
1874       \undef\@gls@restoreat
1875       \gls@defdocnewglossaryentry
1876     \else
1877       \ifnum\@glsxtr@docdefval=3\relax

```

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

```

1878       \@gls@enablesavenonumberlist
1879       \let\gls@checkseeallowed\relax
1880       \let\newglossaryentry\new@atom@glossaryentry
1881       \global\newwrite\@gls@deffile
1882       \immediate\openout\@gls@deffile=\jobname.glsdefs

```

Write all currently defined entries.

```

1883       \forallglsentries{\@glsentry}{\@gls@writedef{\@glsentry}}%
1884     \fi
1885   \fi
1886 }
1887 }
1888 {%
1889   \ifnum\@glsxtr@docdefval=3\relax
1890     \PackageError{glossaries-extra}{Package option
1891       'docdef=\@glsxtr@docdefsetting' requires at least version 4.37

```

```

1892   of the base glossaries.sty package}{%
1893   \fi
1894 }

```

m@glossaryentry

```

1895 \newrobustcmd{\new@atom@glossaryentry}[2]{%
1896   \gls@defglossaryentry{#1}{#2}%
1897   \@gls@writedef{#1}%
1898 }

```

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

```

1899 \let\glstr@orgmakenoidxglossaries\makenoidxglossaries
1900 \renewcommand{\makenoidxglossaries}{%
1901   \@domakeglossaries
1902   {%
1903     \ifdefequal\@glstr@record@setting\@glstr@record@setting@off
1904     {%
1905       \glstr@orgmakenoidxglossaries

```

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

```

1906   \renewcommand{\@do@seeglossary}[2]{%
1907     \@glstr@wrglossmark
1908     \edef\@gls@label{\glsdetoklabel{##1}}%
1909     \protected@write\@auxout{}{%
1910       \string\@gls@reference
1911       {\csname glo@\@gls@label @type\endcsname}%
1912       {\@gls@label}%
1913       {%
1914         \string\glsseeformat##2}%
1915       }%
1916     }%
1917   }%

```

Check for docdefs=restricted:

```

1918   \if@glstrdocdefrestricted

```

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

```

1919   \renewcommand*\@gls@reference}[3]{%
1920     \ifcsundef{\glsref@##1}{\csgdef{\glsref@##1}{}}{}%
1921     \ifinlistcs{##2}{\glsref@##1}%
1922     {}%
1923     {\listcsgadd{\glsref@##1}{##2}}%
1924     \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
1925     {\csgdef{glo@\glsdetoklabel{##2}@loclist}{}}%
1926     {}%
1927     \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
1928   }%

```

```

1929 \else
    Disable document definitions.
1930 \glstrdocdeffalse
1931 \fi
1932 \disable@keys{glossaries-extra}{docdef}%
1933 }%
1934 {%
1935 \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
1936 not permitted\MessageBreak
1937 with record=\glstr@record@setting\space package option}%
1938 {You may only use \string\makenoidxglossaries\space with the
1939 record=off option}%
1940 }%
1941 }%
1942 }

```

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

```

1943 \renewcommand*{\gls@defdocnewglossaryentry}{%
1944 \ifcase\glstr@docdefval
    docdef=false:
1945 \renewcommand*{\newglossaryentry}[2]{%
1946 \PackageError{glossaries-extra}{Glossary entries must
1947 be \MessageBreak defined in the preamble with \MessageBreak
1948 package option ‘docdef=false’\MessageBreak(consider using
1949 ‘docdef=restricted’)}{Move your glossary definitions to
1950 the preamble. You can also put them in a \MessageBreak separate file
1951 and load them with \string\loadglsentries.}%
1952 }%
1953 \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.
1954 \let\gls@checkseeallowed\relax
1955 \let\newglossaryentry\new@glossaryentry
1956 \else
    Restricted mode just needs to allow the see value.
1957 \let\gls@checkseeallowed\relax
1958 \fi
1959 }%

```

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`

```

1960 \newcommand*{\GlsXtrEnableOnTheFly}{%
1961 \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1962 }

```


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

```

1963 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1964   \renewcommand*{\glsdetoklabel}[1]{%
1965     \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1966     {%
1967       \expandafter\detokenize\expandafter{##1}%
1968     }%
1969     {\detokenize{##1}}}%
1970   }%
1971   \@GlsXtrEnableOnTheFly
1972 }
1973 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1974   \expandafter\if\glsbackslash#1%
1975     #3%
1976   \else
1977     #4%
1978   \fi
1979 }

```

`sxtrstarflywarn`

```

1980 \newcommand*{\glsxtrstarflywarn}{%
1981   \GlossariesExtraWarning{Experimental starred version of
1982   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1983   read the warnings in the glossaries-extra user manual)}}%
1984 }

```

`rEnableOnTheFly`

```

1985 \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`

```

1986   \newcommand*{\glsxtrcat}{general}

```

`\glsxtr`

```

1987   \newcommand*{\glsxtr}[1] [] {%
1988     \def\glsxtr@keylist{##1}%
1989     \@glsxtr
1990   }

```

`\@glsxtr`

```

1991   \newcommand*{\@glsxtr}[2] [] {%
1992     \ifglsentryexists{##2}%

```

```

1993   {%
1994     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1995   }%
1996   {%
1997     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1998       description={\nopostdesc},##1}%
1999   }%
2000   \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
2001 }

```

\Glsxtr

```

2002 \newcommand*{\Glsxtr}[1] [] {%
2003   \def\glsxtr@keylist{##1}%
2004   \@Glsxtr
2005 }

```

\@Glsxtr

```

2006 \newcommand*{\@Glsxtr}[2] [] {%
2007   \ifglsentryexists{##2}%
2008   {%
2009     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
2010   }%
2011   {%
2012     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
2013       description={\nopostdesc},##1}%
2014   }%
2015   \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
2016 }

```

\glsxtrpl

```

2017 \newcommand*{\glsxtrpl}[1] [] {%
2018   \def\glsxtr@keylist{##1}%
2019   \@glsxtrpl
2020 }

```

\@glsxtrpl

```

2021 \newcommand*{\@glsxtrpl}[2] [] {%
2022   \ifglsentryexists{##2}%
2023   {%
2024     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
2025   }%
2026   {%
2027     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
2028       description={\nopostdesc},##1}%
2029   }%
2030   \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
2031 }

```

\Glsxtrpl

```

2032 \newcommand*{\Glsxtrpl}[1][]{%
2033 \def\glxtr@keylist{##1}%
2034 \@Glsxtrpl
2035 }

```

\@Glsxtrpl

```

2036 \newcommand*{\@Glsxtrpl}[2][]{%
2037 \ifglstryexists{##2}
2038 {%
2039 \ifblank{##1}{\GlsXtrWarning{##1}{##2}}%
2040 }%
2041 {%
2042 \gls@defglossaryentry{##2}{name={##2},category=\glxtrcat,
2043 description={\nopostdesc},##1}%
2044 }%
2045 \expandafter\Glspl\expandafter[\glxtr@keylist]{##2}%
2046 }

```

\GlsXtrWarning

```

2047 \newcommand*{\GlsXtrWarning}[2]{%
2048 \def\glxtr@optlist{##1}%
2049 \@onelevel@sanitize\glxtr@optlist
2050 \GlossariesExtraWarning{The options '\glxtr@optlist' have
2051 been ignored for entry '##2' as it has already been defined}%
2052 }

```

Disable commands after the glossary:

```

2053 \renewcommand\@printglossary[2]{%
2054 \def\glxtr@printglossopts{##1}%
2055 \@glxtr@orgprintglossary{##1}{##2}%
2056 \def\glxtr{\glxtr@disabledflycommand\glxtr}%
2057 \def\glxtrpl{\glxtr@disabledflycommand\glxtrpl}%
2058 \def\@Glsxtr{\glxtr@disabledflycommand\Glsxtr}%
2059 \def\@Glsxtrpl{\glxtr@disabledflycommand\Glsxtrpl}%
2060 }

```

abledflycommand

```

2061 \newcommand*{\@glxtr@disabledflycommand}[1]{%
2062 \PackageError{glossaries-extra}%
2063 {\string##1\space can't be used after any of the \MessageBreak
2064 glossaries have been displayed}%
2065 {The on-the-fly commands enabled by
2066 \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
2067 before the glossaries. If you want to use any entries \MessageBreak
2068 after any of the glossaries, you must use the standard \MessageBreak
2069 method of first defining the entry and then using the \MessageBreak
2070 entry with commands like \string\gls}%
2071 \@@glxtr@disabledflycommand
2072 }%

```

```
2073 \newcommand*{\@glsxtr@disabledflycommand}[2][\@##2]
```

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

```
2074 \let\GlsXtrEnableOnTheFly\relax
2075 }
2076 \@onlypreamble\GlsXtrEnableOnTheFly
```

1.3.3 Existing Glossary Style Modifications

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

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

```
2077 \newcommand*{\@glsxtr@current@style}{\@glossary@default@style}
```

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

etglossarystyle

```
2078 \renewcommand*{\setglossarystyle}[1]{%
2079   \ifcsundef{@glsstyle@#1}%
2080   {%
2081     \PackageError{glossaries-extra}{Glossary style ‘#1’ undefined}{}%
2082   }%
2083   {%
2084     \csname @glsstyle@#1\endcsname
```

Only set the current style if it exists.

```
2085   \protected@edef\@glsxtr@current@style{#1}%
2086   }%
2087   \ifx\@glossary@default@style\relax
2088     \protected@edef\@glossary@default@style{#1}%
2089   \fi
2090 }
```

In case we have an old version of glossaries:

```
2091 \ifdef\@glossary@default@style
2092 {}
2093 {%
2094   \let\@glossary@default@style\relax
2095 }
```

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

```
2096 \ifdef\glslistdottedwidth
2097 {%
2098   \ifdim\glslistdottedwidth=.5\hsize
2099     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
2100   \AtBeginDocument{%
2101     \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
```

```

2102     \setlength{\glslistdottedwidth}{.5\columnwidth}%
2103     \fi
2104   }%
2105 \fi
2106 }
2107 {}%

```

Similarly for \glsdescwidth:

\glsdescwidth

```

2108 \ifdef\glsdescwidth
2109 {%
2110   \ifdim\glsdescwidth=.6\hsize
2111     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
2112     \AtBeginDocument{%
2113       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
2114         \setlength{\glsdescwidth}{.6\columnwidth}%
2115       \fi
2116     }%
2117   \fi
2118 }
2119 {}%

```

and for \glspagelistwidth:

\glspagelistwidth

```

2120 \ifdef\glspagelistwidth
2121 {%
2122   \ifdim\glspagelistwidth=.1\hsize
2123     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
2124     \AtBeginDocument{%
2125       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
2126         \setlength{\glspagelistwidth}{.1\columnwidth}%
2127       \fi
2128     }%
2129   \fi
2130 }
2131 {}%

```

aryentrynumbers Has the nonumberlist option been used?

```

2132 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
2133 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
2134   \glsnonumberlistfalse
2135   \renewcommand*{\glossaryentrynumbers}[1]{%
2136     \ifglstryentryexists{\glscurrententrylabel}%
2137     {%
2138       \@glsxtrpreloctag
2139       \GlsXtrFormatLocationList{#1}%
2140       \@glsxtrpostloctag
2141       \gls@save@numberlist{#1}%

```

```

2142     }{}%
2143 }%
2144 \else
2145   \glsonumberlisttrue
2146   \renewcommand*{\glossaryentrynumbers}[1]{%
2147     \ifglseentryexists{\glscurrententrylabel}%
2148     {%
2149       \glsave@numberlist{#1}%
2150     }{}%
2151 }%
2152 \fi

```

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

```

2153 \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

```

2154 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
2155   \let\@glstrpreloctag\@glstrpreloctag
2156   \let\@glstrpostloctag\@glstrpostloctag
2157   \renewcommand*{\@glstr@pagetag}{#1}%
2158   \renewcommand*{\@glstr@pagetag}{#2}%
2159   \renewcommand*{\@glstr@savepreloctag}[2]{%
2160     \csgdef{\@glstr@preloctag@##1}{##2}%
2161   }%
2162   \renewcommand*{\@glstr@doloctag}{%
2163     \ifcsundef{\@glstr@preloctag\@glscurrententrylabel}%
2164     {%
2165       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
2166         Rerun required}%
2167     }%
2168     {%
2169       \csuse{\@glstr@preloctag\@glscurrententrylabel}%
2170     }%
2171   }%
2172 }
2173 \@onlypreamble\GlsXtrEnablePreLocationTag

```

glstrpreloctag

```

2174 \newcommand*{\@glstrpreloctag}{%
2175   \let\@glstr@org@delimN\delimN
2176   \let\@glstr@org@delimR\delimR
2177   \let\@glstr@org@glsgignore\glsgignore

```

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

```

2178 \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
2179 \renewcommand*\@delimN{%
2180   \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
2181   \@glxtr@org@delimN}%
2182 \renewcommand*\@delimR{%
2183   \gdef\@glxtr@thisloctag{\@glxtr@pagetag}%
2184   \@glxtr@org@delimR}%
2185 \renewcommand*\@glsglignore}[1]{%
2186   \gdef\@glxtr@thisloctag{\relax}%
2187   \@glxtr@org@glsglignore{##1}}%
2188 \@glxtr@doloctag
2189 }

```

glxtrpreloctag

```

2190 \newcommand*\@glxtrpreloctag{}

```

@glxtr@pagetag

```

2191 \newcommand*\@glxtr@pagetag{}%

```

glxtr@pagetag

```

2192 \newcommand*\@glxtr@pagetag{}%

```

lsxtrpostloctag

```

2193 \newcommand*\@glxtrpostloctag{%
2194   \let\delimN\@glxtr@org@delimN
2195   \let\delimR\@glxtr@org@delimR
2196   \let\glsglignore\@glxtr@org@glsglignore
2197   \protected@write\@auxout{%
2198     {\string\@glxtr@savepreloctag{\glscurrententrylabel}\@glxtr@thisloctag}}%
2199 }

```

lsxtrpostloctag

```

2200 \newcommand*\@glxtrpostloctag{}

```

lsxtr@preloctag

```

2201 \newcommand*\@glxtr@savepreloctag}[2]{%
2202 \protected@write\@auxout{%
2203   \string\providecommand\string\@glxtr@savepreloctag[2]}%

```

glxtr@doloctag

```

2204 \newcommand*\@glxtr@doloctag{}

```

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

```

2205 \renewcommand*\KV@printgloss@nonumberlist}[1]{%
2206 \XKV@plfalse
2207 \XKV@sttrue

```

```

2208 \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
2209 {%
2210   \csname glsnonumberlist\XKV@resa\endcsname
2211   \ifglsnonumberlist
2212     \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
2213   \else
2214     \def\glossaryentrynumbers##1{%
2215       \@glstrpreloctag
2216       \GlsXtrFormatLocationList{##1}%
2217       \@glstrpostloctag
2218       \gls@save@numberlist{##1}}%
2219   \fi
2220 }%
2221 }

```

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.

```

2222 \renewcommand*{\glsentryfmt}{%
2223   \ifglshashshort{\glslabel}{\glsetabbrvfmt{\glscategory{\glslabel}}}%
2224   \glsifregular{\glslabel}%
2225   {\glstrregularfont{\glsgenentryfmt}}%
2226   {%
2227     \ifglshashshort{\glslabel}%
2228     {\glstrabbreviationfont{\glstrgenabbrvfmt}}%
2229     {\glstrregularfont{\glsgenentryfmt}}%
2230   }%
2231 }

```

glstrregularfont Font used for regular entries.

```
2232 \newcommand*{\glstrregularfont}[1]{#1}
```

glstrabbreviationfont Font used for abbreviation entries.

```
2233 \newcommand*{\glstrabbreviationfont}[1]{#1}
```

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

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


```
2234 \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).

```
2235 \@glsxtr@record{#2}{#3}{glslink}%
2236 \glsdoifexists{#3}%
2237 {%
```

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).

```
2238 \let\glsxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
2239 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2240 \def\glscustomtext{#4}%
2241 \@glsxtr@field@linkdefs
2242 #1%
2243 \@gls@link[#2]{#3}{#4}%
2244 \let\ifKV@glslink@hyper\glsxtrorg@ifKV@glslink@hyper
2245 }%
2246 \glspostlinkhook
2247 }
```

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

\@gls@ Save the original definition and redefine.

```
2248 \let\@glsxtr@org@gls@\@gls@
2249 \def\@gls@#1#2{%
2250 \@glsxtr@record{#1}{#2}{glslink}%
2251 \@glsxtr@org@gls@{#1}{#2}%
2252 }%
```

\@glspl@ Save the original definition and redefine.

```
2253 \let\@glsxtr@org@glspl@\@glspl@
2254 \def\@glspl@#1#2{%
2255 \@glsxtr@record{#1}{#2}{glslink}%
2256 \@glsxtr@org@glspl@{#1}{#2}%
2257 }%
```

\@Gls@ Save the original definition and redefine.

```
2258 \let\@glsxtr@org@Gls@\@Gls@
2259 \def\@Gls@#1#2{%
2260 \@glsxtr@record{#1}{#2}{glslink}%
2261 \@glsxtr@org@Gls@{#1}{#2}%
2262 }%
```

\@Glspl@ Save the original definition and redefine.

```
2263 \let\@glsxtr@org@Glspl@\@Glspl@
2264 \def\@Glspl@#1#2{%
2265 \@glsxtr@record{#1}{#2}{glslink}%
```

```

2266 \@glxtr@org@GLspl@{#1}{#2}%
2267 }%

```

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

```

2268 \let\@glxtr@org@GLS@\@GLS@
2269 \def\@GLS@#1#2{%
2270   \@glxtr@record{#1}{#2}{glslink}%
2271   \@glxtr@org@GLS@{#1}{#2}%
2272 }%

```

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

```

2273 \let\@glxtr@org@GLSpl@\@GLSpl@
2274 \def\@GLSpl@#1#2{%
2275   \@glxtr@record{#1}{#2}{glslink}%
2276   \@glxtr@org@GLSpl@{#1}{#2}%
2277 }%

```

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

```

2278 \renewcommand*{\@glsdisp}[3][{}]{%
2279   \@glxtr@record{#1}{#2}{glslink}%
2280   \glsdoifexists{#2}{%
2281     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
2282     \let\glsifplural\@secondoftwo
2283     \let\glscapscase\@firstofthree
2284     \def\glscustomtext{#3}%
2285     \def\glsinsert{}%
2286     \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
2287     \@gl@link[#1]{#2}{\@glo@text}%
2288     \ifKV@glslink@local
2289       \glslocalunset{#2}%
2290     \else
2291       \glsunset{#2}%
2292     \fi
2293   }%
2294   \glspostlinkhook
2295 }

```

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

```

2296 \renewcommand*{\@gl@link}[3][{}]{%
2297   \@glxtr@record{#1}{#2}{glslink}%
2298   \glsdoifexistsordo{#2}%
2299   {%
2300     \let\do@gl@link@checkfirsthyper\relax

```

Post-link hook commands need initialising.

```

2301   \def\glscustomtext{#3}%
2302   \@glxtr@field@linkdefs
2303   \@gl@link[#1]{#2}{#3}%

```

```

2304 }%
2305 {%
2306   \glstextformat{#3}%
2307 }%
2308 \glspostlinkhook
2309 }

```

gsxtrinitwrgloss Set the default if the wrgloss is omitted.

```

2310 \newcommand*{\gsxtrinitwrgloss}{%
2311   \glsifattribute{\glslabel}{wrgloss}{after}%
2312   {%
2313     \gsxtrinitwrglossbeforefalse
2314   }%
2315   {%
2316     \gsxtrinitwrglossbeforetrue
2317   }%
2318 }

```

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

```

2319 \newif\ifgsxtrinitwrglossbefore
2320 \gsxtrinitwrglossbeforetrue

```

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

```

2321 \define@choicekey{glslink}{wrgloss}%
2322 [\@gsxtr@wrglossval\@gsxtr@wrglossnr]%
2323 {before,after}%
2324 {%
2325   \ifcase\@gsxtr@wrglossnr\relax
2326     \gsxtrinitwrglossbeforetrue
2327   \or
2328     \gsxtrinitwrglossbeforefalse
2329   \fi
2330 }

```

```

2331 \define@key{glslink}{thevalue}{\def\@gsxtr@thevalue{#1}}

```

```

2332 \define@key{glslink}{theHvalue}{\def\@gsxtr@theHvalue{#1}}

```

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

```

2333 \define@boolkey{glslink}[gsxtr@]{hyperoutside}[true]{}
2334 \gsxtr@hyperoutsidettrue

```

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

```

2335 \define@key{glslink}{textformat}{%
2336   \ifcsdef{#1}
2337   {%
2338     \letcs{\@gsxtr@local@textformat}{#1}%
2339   }%

```

```

2340  {%
2341    \PackageError{glossaries-extra}{Unknown control sequence name ‘#1’}{}%
2342  }%
2343 }

2344 \define@key{glslink}{prefix}{\def\glolinkprefix{#1}}

nithyperoutside  Set the default if the hyperoutside is omitted.
2345 \newcommand*{\glxtrnithyperoutside}{%
2346   \glsifattribute{\glslabel}{hyperoutside}{false}%
2347   {%
2348     \glxtr@hyperoutsidefalse
2349   }%
2350   {%
2351     \glxtr@hyperoutsidetrue
2352   }%
2353 }

r@inc@linkcount  Does nothing by default.
2354 \newcommand*{\glxtr@inc@linkcount}{}

slinkpresetkeys  User hook performed immediately before options are set. Does nothing by default.
2355 \newcommand*{\glslinkpresetkeys}{}

sXtrExpandedFmt  Helper command that (protected) fully expands second argument and then applies it to the
first, which must be a command that takes a single argument.
2356 \newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
2357   \protected@edef\@glxtr@tmp{#2}%
2358   \expandafter#1\expandafter{\@glxtr@tmp}%
2359 }

tion@counter@or  If in a numbered equation, change the counter to equation. This can be overridden by explic-
itly setting the counter in the optional argument of commands like \gls and \glslink.
2360 \newcommand*{\@glxtr@use@equation@counter}{%
2361   \@glxtr@ifnum@mmode{\def\@gls@counter{equation}}{}}%
2362 }

sxtr@do@autoadd  If \GlsXtrAutoAddOnFormat is used, this will automatically use \glsadd. It’s therefore only
used with \@gls@link not with \glsadd otherwise it could trigger an infinite loop. The ar-
gument indicates the key family (glslink or glossadd).
2363 \newcommand*{\glxtr@do@autoadd}[1]{%

```

AutoAddOnFormat

`\GlsXtrAutoAddOnFormat[⟨label⟩]{⟨format list⟩}{⟨glsadd options⟩}`

If an entry is indexed with the format set to one identified in the comma-separated list, then automatically index it using `\glsadd` with the given options, which may override the current options. Scoping is needed to prevent leakage.

```

2364 \newcommand*{\GlsXtrAutoAddOnFormat}[3][\glslabel]{%
2365   \renewcommand*{\glsxtr@do@autoadd}[1]{%
2366     \begingroup
2367       \protected@edef\@glsxtr@do@autoadd{%
2368         \noexpand\ifstrequal{##1}{\glslink}%
2369         {%
2370           \noexpand\DTLifinlist{\@glsnumberformat}{#2}{\noexpand\glsadd[format={\@glsnumberfor
2371           }%
2372           }}%
2373       }%
2374       \@glsxtr@do@autoadd
2375     \endgroup
2376   }%
2377 }
```

`\@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`.

```

2378 \def\@gls@link[#1]#2#3{%
2379   \leavevmode
2380   \edef\glslabel{\glsdetoklabel{#2}}%
2381   \def\@gls@link@opts{#1}%
2382   \let\@gls@link@label\glslabel
2383   \let\@glsnumberformat\@glsxtr@defaultnumberformat
2384   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
2385   \edef\@gls@type{\csname glo@\glslabel @type\endcsname}%
2386   \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Save current value of `\glolinkprefix`:

```
2387 \let\@glsxtr@org@glolinkprefix\glolinkprefix
```

Initialise `\@glsxtr@local@textformat`

```
2388 \let\@glsxtr@local@textformat\relax
```

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

```

2389 \def\@glsxtr@thevalue{%
2390 \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

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

```
2391 \glsxtrinitwrgloss
```

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

```
2392 \glsxtrinithyperoutside
```

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

```
2393 \@gls@setdefault@glslink@opts
```

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

```

2394 \glxstr@inc@linkcount

```

Check if the equations option has been set (new to v1.37).

```

2395 \if@glxstr@equations
2396 \glxstr@use@equation@counter
2397 \fi

```

As the original definition.

```

2398 \do@glstdisablehyperinlist
2399 \do@glsl@link@checkfirsthyper

```

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

```

2400 \glslinkpresetkeys

```

Set options.

```

2401 \setkeys{glslink}{#1}%

```

Perform auto add if set (new to v1.37)

```

2402 \glxstr@do@autoadd{glslink}%

```

User hook after options are set:

```

2403 \glslinkpostsetkeys

```

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

```

2404 \ifdefempty{\@glxstr@thevalue}%
2405 {%
2406 \glsl@saveentrycounter
2407 }%
2408 {%
2409 \let\theglslentrycounter\@glxstr@thevalue
2410 \def\theHglslentrycounter{\@glxstr@theHvalue}%
2411 }%
2412 \@glsl@setsort{glslabel}%

```

Check if the textformat key has been used.

```

2413 \ifx\@glxstr@local@textformat\relax

```

Check textformat attribute (new to v1.21).

```

2414 \glshasattribute{glslabel}{textformat}%
2415 {%
2416 \edef\@glxstr@attrval{\glsl@getattribute{glslabel}{textformat}}%
2417 \ifcsdef{\@glxstr@attrval}%
2418 {%
2419 \letcs{\@glxstr@textformat}{\@glxstr@attrval}%
2420 }%
2421 {%
2422 \GlossariesExtraWarning{Unknown control sequence name
2423 '\@glxstr@attrval' supplied in textformat attribute
2424 for entry 'glslabel'. Reverting to default \string\glstextformat}%
2425 \let\@glxstr@textformat\glstextformat
2426 }%
2427 }%

```

```

2428     {%
2429     \let\@glsxtr@textformat\glstextformat
2430     }%
2431 \else
2432     \let\@glsxtr@textformat\@glsxtr@local@textformat
2433 \fi

```

Do write if it should occur before the link text:

```

2434 \ifglsxtr@nitrwrglossbefore
2435 \do@wrglossary{#2}%
2436 \fi

```

Do the link text:

```

2437 \ifKV@glslink@hyper
2438 \ifglsxtr@hyperoutside
2439 \glslink{\glolinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
2440 \else
2441 \glsxtr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
2442 \fi
2443 \else
2444 \ifglsxtr@hyperoutside
2445 \glsdonohyperlink{\glolinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
2446 \else
2447 \glsxtr@textformat{\glsdonohyperlink{\glolinkprefix\glslabel}{#3}}%
2448 \fi
2449 \fi

```

Do write if it should occur after the link text:

```

2450 \ifglsxtr@nitrwrglossbefore
2451 \else
2452 \do@wrglossary{#2}%
2453 \fi

```

Restore original value of \glolinkprefix:

```

2454 \let\glolinkprefix\@glsxtr@org@glolinkprefix

```

As the original definition:

```

2455 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
2456 }

```

```

2457 \define@key{glossadd}{thevalue}{\def\@glsxtr@thevalue{#1}}

```

```

2458 \define@key{glossadd}{theHvalue}{\def\@glsxtr@theHvalue{#1}}

```

lsaddpresetkeys

```

2459 \newcommand*{\glsaddpresetkeys}{ }

```

saddpostsetkeys

```

2460 \newcommand*{\glsaddpostsetkeys}{ }

```

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

```
2461 \renewrobustcmd*{\glsadd}[2][\]{%
2462   \glsxtrifinmark
2463   }{%
2464   {%
2465     \@gls@adjustmode
2466     \begingroup
2467     \glsxtr@record{#1}{#2}{glossadd}%
2468     \glsdoifexists{#2}%
2469     {%
2470       \let\@glsnumberformat\@glsxtr@defaultnumberformat
2471       \edef\@gls@counter{\csname glo\@glsdetoklabel{#2}@counter\endcsname}%
2472       \def\@glsxtr@thevalue{%
2473         \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

Implement any default settings (before options are set)

```
2474   \glsaddpresetkeys
2475   \setkeys{glossadd}{#1}%
```

Implement any default settings (after options are set)

```
2476   \glsaddpostsetkeys
2477   \ifdefempty{\@glsxtr@thevalue}%
2478   {%
2479     \@gls@saveentrycounter
2480   }%
2481   {%
2482     \let\theglsentrycounter\@glsxtr@thevalue
2483     \def\theHglentrycounter{\@glsxtr@theHvalue}%
2484   }%
```

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

```
2485   \@gls@setsort{#2}%
```

Ensure that indexing occurs (since that's the point of `\glsadd`). If indexing has been switched off by default, don't want the setting to affect `\glsadd`. The ignored format `\glsignore` can be used for selection without location, but the indexing still needs to be performed.

```
2486   \KV@glslink@noindexfalse
2487   \@do@wrglossary{#2}%
2488   }%
2489   \endgroup
2490   }%
2491 }
```

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

```
2492 \newrobustcmd*{\glsaddeach}[2][\]{%
2493   \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%
2494 }
```

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

```
2495 \newcommand*{\@glsxtr@field@linkdefs}{%
```



```

2496 \let\glxtrifwasfirstuse\@secondoftwo
2497 \let\glisifplural\@secondoftwo
2498 \let\glscapscase\@firstofthree
2499 \let\glinsert\@empty
2500 }

```

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

assignfieldfont

```

2501 \newcommand*\glxtrassignfieldfont}[1]{%
2502   \ifglstryexists{#1}%
2503   {%
2504     \ifglshasshort{#1}%
2505     {%
2506       \glsetabbrvfmt{\glscategory{#1}}%
2507       \glisifregular{#1}%
2508       {\let\@gls@field@font\glxtrregularfont}%
2509       {\let\@gls@field@font\@firstofone}%
2510     }%
2511     {%
2512       \glisifnotregular{#1}%
2513       {\let\@gls@field@font\@firstofone}%
2514       {\let\@gls@field@font\glxtrregularfont}%
2515     }%
2516   }%
2517   {%
2518     \let\@gls@field@font\@gobble
2519   }%
2520 }

```

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

```

2521 \def\@glstext@#1#2[#3]{%
2522   \glxtrassignfieldfont{#2}%
2523   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
2524 }

```

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

```

2525 \def\@GLStext@#1#2[#3]{%
2526   \glxtrassignfieldfont{#2}%
2527   \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
2528   {\@gls@field@font{\@GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
2529 }

```

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

```

2530 \def\@Glstext@#1#2[#3]{%
2531   \glxtrassignfieldfont{#2}%
2532   \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
2533   {\@gls@field@font{\@GLSaccesstext{#2}#3}}%
2534 }

```

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`

```
2535 \newcommand*{\glstrchecknohyperfirst}[1]{%
2536   \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
2537 }
```

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

```
2538 \def\@glsfirst@#1#2[#3]{%
2539   \glstrassignfieldfont{#2}%
    Ensure that \glsfirst honours the nohyperfirst attribute.
2540   \@gls@field@link
2541   [\let\glstrifwasfirstuse\@firstoftwo
2542     \glstrchecknohyperfirst{#2}%
2543   ]{#1}{#2}%
2544   {\@gls@field@font{\glsaccessfirst{#2}#3}}%
2545 }
```

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

```
2546 \def\@Glsfirst@#1#2[#3]{%
2547   \glstrassignfieldfont{#2}%
    Ensure that \@Glsfirst honours the nohyperfirst attribute.
2548   \@gls@field@link
2549   [\let\glstrifwasfirstuse\@firstoftwo
2550     \let\glscapscase\@secondofthree
2551     \glstrchecknohyperfirst{#2}%
2552   ]%
2553   {#1}{#2}{\@gls@field@font{\@Glsaccessfirst{#2}#3}}%
2554 }
```

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

```
2555 \def\@GLSfirst@#1#2[#3]{%
2556   \glstrassignfieldfont{#2}%
    Ensure that \@GLSfirst honours the nohyperfirst attribute.
2557   \@gls@field@link
2558   [\let\glstrifwasfirstuse\@firstoftwo
2559     \let\glscapscase\@thirdofthree
2560     \glstrchecknohyperfirst{#2}%
2561   ]%
2562   {#1}{#2}{\@gls@field@font{\@GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
2563 }
```

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

```
2564 \def\@glsplural@#1#2[#3]{%
2565   \glstrassignfieldfont{#2}%
2566   \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%

```

```

2567     {\@gls@field@font{\glsaccessplural{#2}#3}}%
2568 }

```

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

```

2569 \def\@Glsplural@#1#2[#3]{%
2570   \glstrassignfieldfont{#2}%
2571   \@gls@field@link
2572   [\let\glsifplural\@firstoftwo
2573    \let\glscapscase\@secondofthree
2574   ]%
2575   {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
2576 }

```

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

```

2577 \def\@GLSplural@#1#2[#3]{%
2578   \glstrassignfieldfont{#2}%
2579   \@gls@field@link
2580   [\let\glsifplural\@firstoftwo
2581    \let\glscapscase\@thirdofthree
2582   ]%
2583   {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
2584 }

```

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

```

2585 \def\@glsfirstplural@#1#2[#3]{%
2586   \glstrassignfieldfont{#2}%
2587   \@gls@field@link
2588   [\let\glstrifwasfirstuse\@firstoftwo
2589    \let\glsifplural\@firstoftwo
2590    \glstrchecknohyperfirst{#2}%
2591   ]%
2592   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
2593 }

```

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

```

2594 \def\@Glsfirstplural@#1#2[#3]{%
2595   \glstrassignfieldfont{#2}%
2596   \@gls@field@link
2597   [\let\glstrifwasfirstuse\@firstoftwo
2598    \let\glsifplural\@firstoftwo
2599    \let\glscapscase\@secondofthree
2600    \glstrchecknohyperfirst{#2}%
2601   ]%
2602   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
2603 }

```

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

```
2604 \def\@GLSfirstplural@#1#2[#3]{%
2605   \glstrassignfieldfont{#2}%
      Ensure that \glsfirstplural honours the nohyperfirst attribute.
2606   \@gls@field@link
2607   [\let\glstrifwasfirstuse\@firstoftwo
2608    \let\glsifplural\@firstoftwo
2609    \let\glscapscase\@thirdofthree
2610    \glstrchecknohyperfirst{#2}%
2611   ]%
2612   {#1}{#2}%
2613   {\@gls@field@font{\@GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
2614 }
```

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

```
2615 \def\@glsname@#1#2[#3]{%
2616   \glstrassignfieldfont{#2}%
2617   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessname{#2}#3}}%
2618 }
```

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

```
2619 \def\@Glsname@#1#2[#3]{%
2620   \glstrassignfieldfont{#2}%
2621   \@gls@field@link
2622   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2623   {\@gls@field@font{\@Glsaccessname{#2}#3}}%
2624 }
```

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

```
2625 \def\@GLSname@#1#2[#3]{%
2626   \glstrassignfieldfont{#2}%
2627   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2628   {#1}{#2}%
2629   {\@gls@field@font{\@GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
2630 }
```

\@glsdesc@

```
2631 \def\@glsdesc@#1#2[#3]{%
2632   \glstrassignfieldfont{#2}%
2633   \@gls@field@link{#1}{#2}{\@gls@field@font{\@glsaccessdesc{#2}#3}}%
2634 }
```

\@Glsdesc@ First letter uppercase version.

```
2635 \def\@Glsdesc@#1#2[#3]{%
2636   \glstrassignfieldfont{#2}%
2637   \@gls@field@link
2638   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2639   {\@gls@field@font{\@Glsaccessdesc{#2}#3}}%
2640 }
```

```

\@GLSdesc@ All uppercase version.
2641 \def\@GLSdesc@#1#2[#3]{%
2642   \glstrassignfieldfont{#2}%
2643   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2644   {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
2645 }

@glsglscapscase@ No case-changing version.
2646 \def\@glsglscapscase@#1#2[#3]{%
2647   \glstrassignfieldfont{#2}%
2648   \@gls@field@link
2649   [\let\glscapscase\@secondoftwo
2650   \let\glsifplural\@firstoftwo
2651   ]{#1}{#2}{\@gls@field@font{\GLSaccessdescplural{#2}#3}}%
2652 }

@Glsdescplural@ First letter uppercase version.
2653 \def\@Glsdescplural@#1#2[#3]{%
2654   \glstrassignfieldfont{#2}%
2655   \@gls@field@link
2656   [\let\glscapscase\@secondoftwo
2657   \let\glsifplural\@firstoftwo
2658   ]{#1}{#2}{\@gls@field@font{\GLSaccessdescplural{#2}#3}}%
2659 }

@GLSdescplural@ All uppercase version.
2660 \def\@GLSdesc@#1#2[#3]{%
2661   \glstrassignfieldfont{#2}%
2662   \@gls@field@link
2663   [\let\glscapscase\@thirdoftwo
2664   \let\glsifplural\@firstoftwo
2665   ]%
2666   {#1}{#2}%
2667   {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
2668 }

\@glsymbol@
2669 \def\@glsymbol@#1#2[#3]{%
2670   \glstrassignfieldfont{#2}%
2671   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
2672 }

\@Glsymbol@ First letter uppercase version.
2673 \def\@Glsymbol@#1#2[#3]{%
2674   \glstrassignfieldfont{#2}%
2675   \@gls@field@link
2676   [\let\glscapscase\@secondoftwo]%
2677   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}#3}}%
2678 }

```

\@GLSsymbol@ All uppercase version.

```
2679 \def\@GLSsymbol@#1#2[#3]{%
2680   \glstrassignfieldfont{#2}%
2681   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2682   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
2683 }
```

lssymbolplural@ No case-changing version.

```
2684 \def\@glssymbolplural@#1#2[#3]{%
2685   \glstrassignfieldfont{#2}%
2686   \@gls@field@link
2687   [\let\glscapscase\@secondoftwo
2688   \let\glsifplural\@firstoftwo
2689   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
2690 }
```

lssymbolplural@ First letter uppercase version.

```
2691 \def\@Glssymbolplural@#1#2[#3]{%
2692   \glstrassignfieldfont{#2}%
2693   \@gls@field@link
2694   [\let\glscapscase\@secondoftwo
2695   \let\glsifplural\@firstoftwo
2696   ]{#1}{#2}{\@gls@field@font{\GLsaccesssymbolplural{#2}#3}}%
2697 }
```

LSsymbolplural@ All uppercase version.

```
2698 \def\@GLSsymbol@#1#2[#3]{%
2699   \glstrassignfieldfont{#2}%
2700   \@gls@field@link
2701   [\let\glscapscase\@thirdoftwo
2702   \let\glsifplural\@firstoftwo
2703   ]%
2704   {#1}{#2}%
2705   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
2706 }
```

\@Glsuseri@ First letter uppercase version.

```
2707 \def\@Glsuseri@#1#2[#3]{%
2708   \glstrassignfieldfont{#2}%
2709   \@gls@field@link
2710   [\let\glscapscase\@secondoftwo]{#1}{#2}%
2711   {\@gls@field@font{\GLsentryuseri{#2}#3}}%
2712 }
```

\@GLSuseri@ All uppercase version.

```
2713 \def\@GLSuseri@#1#2[#3]{%
2714   \glstrassignfieldfont{#2}%
2715   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
```

```

2716      {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseri{#2}#3}}}%
2717 }

```

\@Glsuserii@ First letter uppercase version.

```

2718 \def\@Glsuserii@#1#2[#3]{%
2719   \glstrassignfieldfont{#2}%
2720   \@gls@field@link
2721   [\let\glscapscase\@secondoftwo]%
2722   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}}%
2723 }

```

\@GLSuserii@ All uppercase version.

```

2724 \def\@GLSuserii@#1#2[#3]{%
2725   \glstrassignfieldfont{#2}%
2726   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2727   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
2728 }

```

\@Glsuseriii@ First letter uppercase version.

```

2729 \def\@Glsuseriii@#1#2[#3]{%
2730   \glstrassignfieldfont{#2}%
2731   \@gls@field@link
2732   [\let\glscapscase\@secondoftwo]%
2733   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}}%
2734 }

```

\@GLSuseriii@ All uppercase version.

```

2735 \def\@GLSuseriii@#1#2[#3]{%
2736   \glstrassignfieldfont{#2}%
2737   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2738   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
2739 }

```

\@Glsuseriv@ First letter uppercase version.

```

2740 \def\@Glsuseriv@#1#2[#3]{%
2741   \glstrassignfieldfont{#2}%
2742   \@gls@field@link
2743   [\let\glscapscase\@secondoftwo]%
2744   {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}}%
2745 }

```

\@GLSuseriv@ All uppercase version.

```

2746 \def\@GLSuseriv@#1#2[#3]{%
2747   \glstrassignfieldfont{#2}%
2748   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2749   {#1}{#2}%
2750   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
2751 }

```

\@Glsuserv@ First letter uppercase version.

```
2752 \def\@Glsuserv@#1#2[#3]{%
2753   \glxtrassignfieldfont{#2}%
2754   \@gls@field@link
2755   [\let\glscapscase\@secondoftwo]%
2756   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
2757 }
```

\@GLSuserv@ All uppercase version.

```
2758 \def\@GLSuserv@#1#2[#3]{%
2759   \glxtrassignfieldfont{#2}%
2760   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2761   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
2762 }
```

\@Glsuservi@ First letter uppercase version.

```
2763 \def\@Glsuservi@#1#2[#3]{%
2764   \glxtrassignfieldfont{#2}%
2765   \@gls@field@link
2766   [\let\glscapscase\@secondoftwo]%
2767   {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
2768 }
```

\@GLSuservi@ All uppercase version.

```
2769 \def\@GLSuservi@#1#2[#3]{%
2770   \glxtrassignfieldfont{#2}%
2771   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
2772   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
2773 }
```

Commands like \acrshort already set \glsifplural, but they don't set \glxtrifwasfirstuse so they need adjusting. These commands shouldn't be used with \newabbreviation, but the redefinitions below allow for users reverting \newacronym back to its base definition.

base@acrcmd@warn Warn user that they need to use to new abbreviation commands.

```
2774 \newcommand*{\@@glxtr@base@acrcmd@warn}[2]{%
2775   \GlossariesExtraWarning{Base acronym command \string#1\space
2776     should not be used with new abbreviation definitions. Use
2777     \string#2\space instead}%
2778 }
```

glxtr@base@acrcmd Warn user that they need to use to new abbreviation commands.

```
2779 \let\@glxtr@base@acrcmd\@@glxtr@base@acrcmd@warn
```

\@acrshort No case change.

```
2780 \def\@acrshort#1#2[#3]{%
2781   \@glxtr@base@acrcmd\acrshort\glxtrshort
2782   \glstdoifexists{#2}%
```



```

2783 {%
2784   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2785   \let\glxtrifwasfirstuse\@secondoftwo
2786   \let\gl@sifplural\@secondoftwo
2787   \let\glscapscase\@firstofthree
2788   \let\gl$insert\@empty
2789   \def\glscustomtext{%
2790     \acronymfont{\gl@saccesssshort{#2}}#3%
2791   }%
2792   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2793 }%
2794 \glspostlinkhook
2795 }

```

\@Acrshort First letter uppercase.

```

2796 \def\@Acrshort#1#2[#3]{%
2797   \@glxtr@base@acrcmd\Acrshort\Glsxtrshort
2798   \gl@sdoifexists{#2}%
2799   {%
2800     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2801     \let\glxtrifwasfirstuse\@secondoftwo
2802     \let\gl@sifplural\@secondoftwo
2803     \let\glscapscase\@secondofthree
2804     \let\gl$insert\@empty
2805     \def\glscustomtext{%
2806       \acronymfont{\Glsaccesssshort{#2}}#3%
2807     }%
2808     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2809   }%
2810   \glspostlinkhook
2811 }

```

\@ACRshort All uppercase.

```

2812 \def\@ACRshort#1#2[#3]{%
2813   \@glxtr@base@acrcmd\ACRshort\GLSxtrshort
2814   \gl@sdoifexists{#2}%
2815   {%
2816     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2817     \let\glxtrifwasfirstuse\@secondoftwo
2818     \let\gl@sifplural\@secondoftwo
2819     \let\glscapscase\@thirdofthree
2820     \let\gl$insert\@empty
2821     \def\glscustomtext{%
2822       \mfirstucMakeUppercase{\acronymfont{\gl@saccesssshort{#2}}#3}%
2823     }%
2824     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2825   }%
2826   \glspostlinkhook
2827 }

```

\@acrshortpl No case change.

```
2828 \def\@acrshortpl#1#2[#3]{%
2829   \@glsxtr@base@acrcmd\acrshortpl\glsxtrshortpl
2830   \glsdoifexists{#2}%
2831   {%
2832     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2833     \let\glsxtrifwasfirstuse\@secondoftwo
2834     \let\glsifplural\@firstoftwo
2835     \let\gls caps case\@firstofthree
2836     \let\glsinsert\@empty
2837     \def\gls custom text{%
2838       \acronymfont{\gls access shortpl{#2}}#3%
2839     }%
2840     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2841   }%
2842   \gls post link hook
2843 }
```

\@Acrshortpl First letter uppercase.

```
2844 \def\@Acrshortpl#1#2[#3]{%
2845   \@glsxtr@base@acrcmd\Acrshortpl\Glsxtrshortpl
2846   \glsdoifexists{#2}%
2847   {%
2848     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2849     \let\glsxtrifwasfirstuse\@secondoftwo
2850     \let\glsifplural\@firstoftwo
2851     \let\gls caps case\@secondofthree
2852     \let\glsinsert\@empty
2853     \def\gls custom text{%
2854       \acronymfont{\Gls access shortpl{#2}}#3%
2855     }%
2856     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2857   }%
2858   \gls post link hook
2859 }
```

\@ACRshortpl All uppercase.

```
2860 \def\@ACRshortpl#1#2[#3]{%
2861   \@glsxtr@base@acrcmd\ACRshortpl\GLSxtrshortpl
2862   \glsdoifexists{#2}%
2863   {%
2864     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2865     \let\glsxtrifwasfirstuse\@secondoftwo
2866     \let\glsifplural\@firstoftwo
2867     \let\gls caps case\@thirdofthree
2868     \let\glsinsert\@empty
2869     \def\gls custom text{%
2870       \mfirstucMakeUppercase{\acronymfont{\gls access shortpl{#2}}#3}%
2871     }%
2872   }
```

```

2872 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2873 }%
2874 \glspostlinkhook
2875 }

```

\@acrlong No case change.

```

2876 \def\@acrlong#1#2[#3]{%
2877 \glsxtr@base@acrcmd\acrlong\glsxtrlong
2878 \glsdoifexists{#2}%
2879 {%
2880 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2881 \let\glsxtrifwasfirstuse\@secondoftwo
2882 \let\glsifplural\@secondoftwo
2883 \let\glscapscase\@firstofthree
2884 \let\glsinsert\@empty
2885 \def\glscustomtext{%
2886 \acronymfont{\glsaccesslong{#2}}#3%
2887 }%
2888 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2889 }%
2890 \glspostlinkhook
2891 }

```

\@Acrlong First letter uppercase.

```

2892 \def\@Acrlong#1#2[#3]{%
2893 \glsxtr@base@acrcmd\Acrlong\Glsxtrlong
2894 \glsdoifexists{#2}%
2895 {%
2896 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2897 \let\glsxtrifwasfirstuse\@secondoftwo
2898 \let\glsifplural\@secondoftwo
2899 \let\glscapscase\@secondofthree
2900 \let\glsinsert\@empty
2901 \def\glscustomtext{%
2902 \acronymfont{\Glsaccesslong{#2}}#3%
2903 }%
2904 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2905 }%
2906 \glspostlinkhook
2907 }

```

\@ACRlong All uppercase.

```

2908 \def\@ACRlong#1#2[#3]{%
2909 \glsxtr@base@acrcmd\ACRlong\GLSxtrlong
2910 \glsdoifexists{#2}%
2911 {%
2912 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2913 \let\glsxtrifwasfirstuse\@secondoftwo
2914 \let\glsifplural\@secondoftwo

```

```

2915 \let\glscapscase\@thirdofthree
2916 \let\glsinsert\@empty
2917 \def\glscustomtext{%
2918     \mfirstucMakeUppercase{\acronymfont{\glsaccesslong{#2}}#3}%
2919 }%
2920 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2921 }%
2922 \glspostlinkhook
2923 }

```

\@acrlongpl No case change.

```

2924 \def\@acrlongpl#1#2[#3]{%
2925     \@glsxtr@base@acrcmd\acrlongpl\glsxtrlongpl
2926     \glsdoifexists{#2}%
2927     {%
2928         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2929         \let\glsxtrifwasfirstuse\@secondoftwo
2930         \let\glsifplural\@firstoftwo
2931         \let\glscapscase\@firstofthree
2932         \let\glsinsert\@empty
2933         \def\glscustomtext{%
2934             \acronymfont{\glsaccesslongpl{#2}}#3%
2935         }%
2936         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2937     }%
2938     \glspostlinkhook
2939 }

```

\@Acrlongpl First letter uppercase.

```

2940 \def\@Acrlongpl#1#2[#3]{%
2941     \@glsxtr@base@acrcmd\Acrlongpl\Glsxtrlongpl
2942     \glsdoifexists{#2}%
2943     {%
2944         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
2945         \let\glsxtrifwasfirstuse\@secondoftwo
2946         \let\glsifplural\@firstoftwo
2947         \let\glscapscase\@secondofthree
2948         \let\glsinsert\@empty
2949         \def\glscustomtext{%
2950             \acronymfont{\Glsaccesslongpl{#2}}#3%
2951         }%
2952         \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2953     }%
2954     \glspostlinkhook
2955 }

```

\@ACRlongpl All uppercase.

```

2956 \def\@ACRlongpl#1#2[#3]{%
2957     \@glsxtr@base@acrcmd\ACRlongpl\GLSxtrlongpl

```

```

2958 \glsdoifexists{#2}%
2959 {%
2960   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
2961   \let\glxtrifwasfirstuse\@secondoftwo
2962   \let\gl@ifplural\@firstoftwo
2963   \let\gl@scapscase\@thirdofthree
2964   \let\gl@insert\@empty
2965   \def\glscustomtext{%
2966     \mfirstucMakeUppercase{\acronymfont{\gl@saccesslongpl{#2}}{#3}}%
2967   }%
2968   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
2969 }%
2970 \glspostlinkhook
2971 }

```

The full formats use the internal long and short commands (such as `\@acrshort` and `\@acrlong`). Therefore they don't need adjustments, but they do need clearer warnings. This means three warnings per use (once for the full command and once each for the short and long commands), but at least this way the most important warning (replace `\acrfull` with `\glxtrfull` etc) is present.

`\@acrfull`

```

2972 \def\@acrfull#1#2[#3]{%
2973   \@glxtr@base@acrcmd\acrfull\glxtrfull
2974   \acrfullfmt{#1}{#2}{#3}%
2975 }

```

`\@Acrfull`

```

2976 \def\@Acrfull#1#2[#3]{%
2977   \@glxtr@base@acrcmd\Acrfull\Glsxtrfull
2978   \Acrfullfmt{#1}{#2}{#3}%
2979 }

```

`\@ACRfull`

```

2980 \def\@ACRfull#1#2[#3]{%
2981   \@glxtr@base@acrcmd\ACRfull\GLSxtrfull
2982   \ACRfullfmt{#1}{#2}{#3}%
2983 }

```

`\@acrfullpl`

```

2984 \def\@acrfullpl#1#2[#3]{%
2985   \@glxtr@base@acrcmd\acrfullpl\glxtrfullpl
2986   \acrfullplfmt{#1}{#2}{#3}%
2987 }

```

`\@Acrfullpl`

```

2988 \def\@Acrfullpl#1#2[#3]{%
2989   \@glxtr@base@acrcmd\Acrfullpl\Glsxtrfullpl

```

```

2990 \Acrfullplfmt{#1}{#2}{#3}%
2991 }

```

\@ACRfullpl

```

2992 \def\@ACRfullpl#1#2[#3]{%
2993   \@glsxtr@base@acrcmd\ACRfullpl\GLSxtrfullpl
2994   \ACRfullplfmt{#1}{#2}{#3}%
2995 }

```

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

\@glsaddkey

```

2996 \renewcommand*{\@glsaddkey}[7]{%
2997   \key@ifundefined{glossentry}{#1}%
2998   {%
2999     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
3000     \appto\@gls@keymap{,{#1}{#1}}%
3001     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
3002     \appto\@newglossaryentryposthook{%
3003       \letcs{@glo@tmp}{@glo@#1}%
3004       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
3005     }%
3006     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
3007     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

3008   \ifcsdef{@gls@user@#1@}%
3009   {%
3010     \PackageError{glossaries}%
3011     {Can't define '\string#5' as helper command
3012     '\expandafter\string\csname @gls@user@#1\endcsname' already
3013     exists}%
3014     {}%
3015   }%
3016   {%
3017     \expandafter\newcommand\expandafter*\expandafter
3018     {\csname @gls@user@#1\endcsname}[2][ ]{%
3019       \new@ifnextchar[%
3020         {\csuse{@gls@user@#1@}{##1}{##2}}%
3021         {\csuse{@gls@user@#1@}{##1}{##2}[ ]}}%
3022     \csdef{@gls@user@#1@}##1##2[##3]{%
3023       \@gls@field@link{##1}{##2}{#3{##2}##3}%
3024     }%
3025     \newrobustcmd*{#5}{%
3026       \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
3027     }%

```

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

```

3028   \ifcsdef{@Gls@user@#1@}%

```

```

3029  {%
3030      \PackageError{glossaries}%
3031      {Can't define '\string#6' as helper command
3032      '\expandafter\string\csname @Gls@user@#1@\endcsname' already
3033      exists}%
3034  }%
3035  }%
3036  {%
3037      \expandafter\newcommand\expandafter*\expandafter
3038      {\csname @Gls@user@#1@\endcsname}[2][\{%
3039          \new@ifnextchar[%
3040              {\csuse{@Gls@user@#1@}{##1}{##2}}%
3041              {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
3042      \csdef{@Gls@user@#1@}##1##2[##3]{%
3043          \@gls@field@link[\let\glscapscase\@secondofthree]%
3044          {##1}{##2}{#4{##2}##3}%
3045      }%
3046      \newrobustcmd*{#6}{%
3047          \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
3048      }%

```

Finally the all caps version (modified):

```

3049  \ifcsdef{@GLS@user@#1@}%
3050  {%
3051      \PackageError{glossaries}%
3052      {Can't define '\string#7' as helper command
3053      '\expandafter\string\csname @GLS@user@#1@\endcsname' already
3054      exists}%
3055  }%
3056  }%
3057  {%
3058      \expandafter\newcommand\expandafter*\expandafter
3059      {\csname @GLS@user@#1@\endcsname}[2][\{%
3060          \new@ifnextchar[%
3061              {\csuse{@GLS@user@#1@}{##1}{##2}}%
3062              {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
3063      \csdef{@GLS@user@#1@}##1##2[##3]{%
3064          \@gls@field@link[\let\glscapscase\@thirdofthree]%
3065          {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
3066      }%
3067      \newrobustcmd*{#7}{%
3068          \expandafter\@gls@hyp@opt\csname @GLS@user@#1@\endcsname}%
3069      }%
3070  }%
3071  {%
3072      \PackageError{glossaries-extra}{Key '#1' already exists}{}%
3073  }%
3074 }

```

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

```
3075 \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.

```
3076 \let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
3077 \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 to automatically set the value for the commands that change the first use flag. The other commands should set \glsxtrifwasfirstuse to \@secondoftwo (which is done in \@glsxtr@field@linkdefs).

```
3078 \ifglsused{\glslabel}%
3079 {\let\glsxtrifwasfirstuse\@secondoftwo}
3080 {\let\glsxtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```
3081 \edef\glscategorylabel{\glscategory{\glslabel}}%
3082 \ifglsused{\glslabel}%
3083 {%
3084   \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
3085   {\KV@glslink@hyperfalse}{}}%
3086 }%
3087 {%
3088   \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
3089   {\KV@glslink@hyperfalse}{}}%
3090 }%
3091 \glslinkcheckfirsthyperhook
3092 }

```

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.

```
3093 \ifdef\do@glstdisablehyperinlist
3094 {%
3095   \let\@glsxtr@do@glstdisablehyperinlist\do@glstdisablehyperinlist
3096   \renewcommand*{\do@glstdisablehyperinlist}{%
3097     \@glsxtr@do@glstdisablehyperinlist
3098     \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}}%
3099   }
3100 }
3101 {}

```

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

```
3102 \define@boolkey{glslink}{noindex}[true]{}
3103 \KV@glslink@noindexfalse

```

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

lt@glslink@opts

```
3104 \ifdef \@gls@setdefault@glslink@opts
3105 {
3106   \renewcommand*{\@gls@setdefault@glslink@opts}{%
3107     \KV@glslink@noindexfalse
3108     \@glxtrsetaliasnoindex
3109   }
3110 }
3111 {
3112   Not defined so prepend it to \do@glstdisablehyperinlist to achieve the same effect.
3113   \newcommand*{\@gls@setdefault@glslink@opts}{%
3114     \KV@glslink@noindexfalse
3115     \@glxtrsetaliasnoindex
3116   }
3117   \preto\do@glstdisablehyperinlist{\@gls@setdefault@glslink@opts}
```

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.)

```
3118 \providecommand*{\glxtrsetaliasnoindex}{%
3119   \KV@glslink@noindextrue
3120 }
```

setaliasnoindex

```
3121 \newcommand*{\@glxtrsetaliasnoindex}{%
3122   \glxtrifhasfield{alias}{\glslabel}%
3123   {%
3124     \let\glxtrindexaliased\@glxtrindexaliased
3125     \glxtrsetaliasnoindex
3126     \let\glxtrindexaliased\@no@glxtrindexaliased
3127   }%
3128   {%
3129 }
```

xtrindexaliased

```
3130 \newcommand{\@glxtrindexaliased}{%
3131   \ifKV@glslink@noindex
3132   \else
3133     \begingroup
3134     \let\@glsnumberformat\@glxtr@defaultnumberformat
3135     \edef\@gls@counter{\csname glo@\glstdetoklabel{\glslabel}@counter\endcsname}%
3136     \glxtr@saveentrycounter
3137     @@do@wrglossary{\glxtralias{\glslabel}}%
3138     \endgroup
3139   \fi
3140 }
```

xtrindexaliased

```
3141 \newcommand{\@no@glxtrindexaliased}{%
3142   \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
3143   not permitted outside definition of \string\glxtrsetaliasnoindex}%
3144   }%
3145 }
```

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

```
3146 \let\glxtrindexaliased\@no@glxtrindexaliased
```

tDefaultGlsOpts Set the default options for \glslink etc.

```
3147 \newcommand*\GlsXtrSetDefaultGlsOpts}[1]{%
3148   \renewcommand*\@glslsetdefault@glslink@opts{%
3149     \setkeys{glslink}{#1}%
3150     \@glxtrsetaliasnoindex
3151   }%
3152 }
```

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

```
3153 \newcommand*\glxtrifindexing}[2]{%
3154   \ifKV@glslink@noindex #2\else #1\fi
3155 }
```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```
3156 \renewcommand*\glswriteentry}[2]{%
3157   \glxtrifindexing
3158   {%
3159     \ifglslindexonlyfirst
3160       \ifglslused{#1}
3161       {\glxtrdoautoindexname{#1}{dualindex}}%
3162       {#2}%
3163     \else
3164       \glslifattribute{#1}{indexonlyfirst}{true}%
3165       {%
3166         \ifglslused{#1}%
3167         {\glxtrdoautoindexname{#1}{dualindex}}%
3168         {#2}%
3169       }%
3170       {#2}%
3171     \fi
3172   }%
3173   {%
3174 }
```

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

```
3175 \appto\@do@@wrglossary{\@glxtr@do@@wrindex
3176   \glxtrdowrglossaryhook{\@glsl@label}%
3177 }
```

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

Similarly for the “noidx” version:

`s@noidxglossary`

```
3178 \appto\gls@noidxglossary{\@glsxtr@do@@wrindex
3179   \glsxtrdowrglossaryhook{\@gls@label}}%
3180 }
```

`xtr@do@@wrindex`

```
3181 \newcommand*{\@glsxtr@do@@wrindex}{%
3182   \glsxtrdoautoindexname{\@gls@label}{dualindex}}%
3183 }
```

`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.)

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

`gls@alt@hyp@opt`

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

```
3185 \newcommand*{\@gls@alt@hyp@opt}[1]{%
3186   \let\glslinkvar\@firstofthree
3187   \let\@gls@hyp@opt@cs#1\relax
3188   \@ifstar{\s@gls@hyp@opt}%
3189   {\@ifnextchar+%
3190     {\@firstoftwo{\p@gls@hyp@opt}}%
3191     {%
3192       \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
3193       {\@firstoftwo{\@alt@gls@hyp@opt}}%
3194       {#1}%
3195     }}%
3196 }%
3197 }
```

`alt@gls@hyp@opt`

User version

```
3198 \newcommand*{\@alt@gls@hyp@opt}[1] [] {%
3199   \let\glslinkvar\@firstofthree
3200   \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

`lt@hyp@opt@char`

Contains the character used as the command modifier.

```
3201 \newcommand*{\@gls@alt@hyp@opt@char}{{}
```

`lt@hyp@opt@keys`

Contains the option list used as the command modifier.

```
3202 \newcommand*{\@gls@alt@hyp@opt@keys}{{}
```

`rSetAltModifier`

```
3203 \newcommand*{\GlsXtrSetAltModifier}[2]{%
3204   \let\@gls@hyp@opt\@gls@alt@hyp@opt
```

Check that the supplied character isn't + or *

```

3205 \ifstrequal{#1}{+}%
3206 {\PackageError{glossaries-extra}%
3207 {Can't use '#1' as modifier (it's already in use)}}}%
3208 {%
3209 \ifstrequal{#1}{*}%
3210 {\PackageError{glossaries-extra}%
3211 {Can't use '#1' as modifier (it's already in use)}}}%
3212 {}%
3213 }%
3214 \def\@gls@alt@hyp@opt@char{#1}%
3215 \def\@gls@alt@hyp@opt@keys{#2}%
3216 \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
3217 {}%
3218 {%

```

Let **bib2gls** know the modifier.

```

3219 \protected@write\@auxout{}\string\providecommand{\string\@glsxtr@altmodifier}[1]{}%
3220 \protected@write\@auxout{}\string\@glsxtr@altmodifier{#1}}%
3221 }%
3222 }

```

org@dohyperlink

```
3223 \let\glsxtr@org@dohyperlink\glsdohyperlink
```

glsnavhyperlink

Since `\glsnavhyperlink` uses `\@glslink`, it's necessary to patch it uses `\glsdohyperlink` instead of `\glsxtrdohyperlink`. The simplest way to achieve this is to locally let `\glsxtrdohyperlink` to `\glsdohyperlink`.

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

```

3224 \ifdef\glsnavhyperlink
3225 {
3226 \renewcommand*{\glsnavhyperlink}[3][\@glo@type]{%
3227 \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%

```

Scope:

```

3228 {%
3229 \let\glsxtrdohyperlink\glsxtr@org@dohyperlink
3230 \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
3231 }%
3232 }%
3233 }
3234 {}

```

The redefinition of `\glsdohyperlink` has been causing problems so introduce a new command instead.

sxtrdohyperlink

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.

```

3235 \newcommand*{\glxtrdohyperlink}[2]{%
3236   \glshasattribute{\glslabel}{targeturl}%
3237   {%
3238     \glshasattribute{\glslabel}{targetname}%
3239     {%
3240       \glshasattribute{\glslabel}{targetcategory}%
3241       {%
3242         \hyperref{\glsggetattribute{\glslabel}{targeturl}}{%
3243           {\glsggetattribute{\glslabel}{targetcategory}}}%
3244         {\glsggetattribute{\glslabel}{targetname}}}%
3245         {\glxtrprotectlinks#2}}%
3246       }%
3247     {%
3248       \hyperref{\glsggetattribute{\glslabel}{targeturl}}{%
3249         }%
3250       {\glsggetattribute{\glslabel}{targetname}}}%
3251       {\glxtrprotectlinks#2}}%
3252     }%
3253   }%
3254   {%
3255     \href{\glsggetattribute{\glslabel}{targeturl}}{%
3256       {\glxtrprotectlinks#2}}%
3257     }%
3258   }%
3259   {%

```

Check for alias.

```

3260   \glsgfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
3261   \ifdefined\gloaliaslabel
3262   {%
3263     \glxtrhyperlink{#1}{\glxtrprotectlinks#2}}%
3264   }%
3265   {%

```

Redirect link to the alias target.

```

3266   \glxtrhyperlink
3267   {\glolinkprefix\glsgdetoklabel{\gloaliaslabel}}%
3268   {\glxtrprotectlinks#2}}%
3269   }%
3270 }%
3271 }

```

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

```

3272 \ifdefined\glsshowtarget
3273 {
3274   \newcommand{\glxtrhyperlink}[2]{%
3275     \@glsshowtarget{#1}%

```

```

3276     \hyperlink{#1}{#2}%
3277 }%
3278 }
3279 {
3280   \newcommand{\glstrhyperlink}[2]{\hyperlink{#1}{#2}}%
3281 }

```

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

```

3282 \renewrobustcmd*{\glshyperlink}[2][\glstrytext{\@glo@label}]{%
3283   \glsoifexists{#2}%
3284   {%
3285     \def\@glo@label{#2}%
3286     {\edef\glslabel{#2}%
3287       \@glslink{\glolinkprefix\glslabel}{#1}}%
3288   }%
3289 }

```

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

```

3290 \renewcommand{\glisablehyper}{%
3291   \KV@glslink@hyperfalse
3292   \def\@glslink{\glsdonohyperlink}%
3293   \let\@glstarget\@secondoftwo
3294 }

```

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

```

3295 \renewcommand{\glsenablehyper}{%
3296   \KV@glslink@hypertrue
3297   \def\@glslink{\glstrdohyperlink}%
3298   \def\@glstarget{\glsdohypertarget}%
3299 }

```

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

```

3300 \def\glsdonohyperlink#1#2{{\glstrprotectlinks #2}}

```

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

```

3301 \ifcsundef{hyperlink}%
3302 {%
3303   \def\@glslink{\glsdonohyperlink}
3304 }%

```

```

3305 {%
3306   \def\@glslink{\glsxtrdohyperlink}
3307 }

```

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

```

3308 \newcommand*{\glsxtrprotectlinks}{%
3309   \KV@glslink@hyperfalse
3310   \KV@glslink@noindextrue
3311   \let\@gls@\@glsxtr@p@text@
3312   \let\@Gls@\@Glsxtr@p@text@
3313   \let\@GLS@\@GLSxtr@p@text@
3314   \let\@glspl@\@glsxtr@p@plural@
3315   \let\@Glspl@\@Glsxtr@p@plural@
3316   \let\@GLSpl@\@GLSxtr@p@plural@
3317   \let\@glsxtrshort@\@glsxtr@p@short@
3318   \let\@Glsxtrshort@\@Glsxtr@p@short@
3319   \let\@GLSxtrshort@\@GLSxtr@p@short@
3320   \let\@glsxtrlong@\@glsxtr@p@long@
3321   \let\@Glsxtrlong@\@Glsxtr@p@long@
3322   \let\@GLSxtrlong@\@GLSxtr@p@long@
3323   \let\@glsxtrshortpl@\@glsxtr@p@shortpl@
3324   \let\@Glsxtrshortpl@\@Glsxtr@p@shortpl@
3325   \let\@GLSxtrshortpl@\@GLSxtr@p@shortpl@
3326   \let\@glsxtrlongpl@\@glsxtr@p@longpl@
3327   \let\@Glsxtrlongpl@\@Glsxtr@p@longpl@
3328   \let\@GLSxtrlongpl@\@GLSxtr@p@longpl@
3329   \let\@acrshort@\@glsxtr@p@acrshort@
3330   \let\@Acrshort@\@Glsxtr@p@acrshort@
3331   \let\@ACRshort@\@GLSxtr@p@acrshort@
3332   \let\@acrshortpl@\@glsxtr@p@acrshortpl@
3333   \let\@Acrshortpl@\@Glsxtr@p@acrshortpl@
3334   \let\@ACRshortpl@\@GLSxtr@p@acrshortpl@
3335   \let\@acrlong@\@glsxtr@p@acrlong@
3336   \let\@Acrlong@\@Glsxtr@p@acrlong@
3337   \let\@ACRlong@\@GLSxtr@p@acrlong@
3338   \let\@acrlongpl@\@glsxtr@p@acrlongpl@
3339   \let\@Acrlongpl@\@Glsxtr@p@acrlongpl@
3340   \let\@ACRlongpl@\@GLSxtr@p@acrlongpl@
3341 }

```

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

`@glsxtr@p@text@`

```

3342 \def\@glsxtr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}

```

`@Glsxtr@p@text@`

```

3343 \def\@Glsxtr@p@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}

```

```

@GLSxtr@p@text@
3344 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}}

lsxtr@p@plural@
3345 \def\@glsxtr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}}

lsxtr@p@plural@
3346 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}}

LSxtr@p@plural@
3347 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}}

glsxtr@p@short@
3348 \def\@glsxtr@p@short@#1#2[#3]{%
3349 {%
3350   \glssetabbrvfmt{\glscategory{#2}}%
3351   \glsabbrvfont{\glsentryshort{#2}}#3%
3352 }%
3353 }

Glsxtr@p@short@
3354 \def\@Glsxtr@p@short@#1#2[#3]{%
3355 {%
3356   \glssetabbrvfmt{\glscategory{#2}}%
3357   \glsabbrvfont{\Glsentryshort{#2}}#3%
3358 }%
3359 }

GLSxtr@p@short@
3360 \def\@GLSxtr@p@short@#1#2[#3]{%
3361 {%
3362   \glssetabbrvfmt{\glscategory{#2}}%
3363   \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshort{#2}}#3}%
3364 }%
3365 }

sxtr@p@shortpl@
3366 \def\@glsxtr@p@shortpl@#1#2[#3]{%
3367 {%
3368   \glssetabbrvfmt{\glscategory{#2}}%
3369   \glsabbrvfont{\glsentryshortpl{#2}}#3%
3370 }%
3371 }

sxtr@p@shortpl@
3372 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
3373 {%
3374   \glssetabbrvfmt{\glscategory{#2}}%

```



```

3375 \glsabbrvfont{\Glsentryshortpl{#2}}#3%
3376 }%
3377 }

```

Sxtr@p@shortpl@

```

3378 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
3379   {%
3380     \glssetabbrvfmt{\glscategory{#2}}%
3381     \mfirstucMakeUppercase{\glsabbrvfont{\Glsentryshortpl{#2}}#3}%
3382   }%
3383 }

```

@glxtr@p@long@

```

3384 \def\@glxtr@p@long@#1#2[#3]{\Glsentrylong{#2}#3}

```

@Glsxtr@p@long@

```

3385 \def\@Glsxtr@p@long@#1#2[#3]{\Glsentrylong{#2}#3}

```

@GLSxtr@p@long@

```

3386 \def\@GLSxtr@p@long@#1#2[#3]{%
3387   {\mfirstucMakeUppercase{\glsfont{\Glsentrylong{#2}}#3}}

```

lsxtr@p@longpl@

```

3388 \def\@glxtr@p@longpl@#1#2[#3]{\Glsentrylongpl{#2}#3}

```

lsxtr@p@longpl@

```

3389 \def\@Glsxtr@p@longpl@#1#2[#3]{\glsfont{\Glsentrylongpl{#2}}#3}

```

LSxtr@p@longpl@

```

3390 \def\@GLSxtr@p@longpl@#1#2[#3]{%
3391   {\mfirstucMakeUppercase{\glsfont{\Glsentrylongpl{#2}}#3}}

```

xtr@p@acrshort@

```

3392 \def\@glxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}

```

xtr@p@acrshort@

```

3393 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}

```

xtr@p@acrshort@

```

3394 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
3395   {\mfirstucMakeUppercase{\acronymfont{\Glsentryshort{#2}}#3}}

```

r@p@acrshortpl@

```

3396 \def\@glxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}

```

r@p@acrshortpl@

```

3397 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}

```

```

r@p@acrshortpl@
3398 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
3399   {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}#3}}}}

```

```

s@p@acrlong@
3400 \def\@glsxtr@p@acrlong@#1#2[#3]{\glsentrylong{#2}#3}}

```

```

s@p@acrlong@
3401 \def\@GLSxtr@p@acrlong@#1#2[#3]{\Glsentrylong{#2}#3}}

```

```

S@p@acrlong@
3402 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
3403   {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}}

```

```

tr@p@acrlongpl@
3404 \def\@glsxtr@p@acrlongpl@#1#2[#3]{\glsentrylongpl{#2}#3}}

```

```

tr@p@acrlongpl@
3405 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{\Glsentrylongpl{#2}#3}}

```

```

tr@p@acrlongpl@
3406 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
3407   {\mfirstucMakeUppercase{\glsentrylongpl{#2}#3}}}}

```

Commands to minimise conflict.

```

\@glsxtrp@opt
3408 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}

```

```

\glsxtrsetpopts  Used in glossary to switch hyperlinks on for the \glsxtrp type of commands.
3409 \newcommand*{\glsxtrsetpopts}[1]{%
3410   \renewcommand*{\@glsxtrp@opt}{#1}%
3411 }

```

```

\glossxtrsetpopts  Used in glossary to switch hyperlinks on for the \glsxtrp type of commands.
3412 \newcommand*{\glossxtrsetpopts}{%
3413   \glsxtrsetpopts{noindex}%
3414 }

```

```

\@@glsxtrp
3415 \newrobustcmd*{\@@glsxtrp}[2]{%
  Add scope.
3416   {%
3417     \let\glspostlinkhook\relax
3418     \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
3419   }%
3420 }

```

\@glsxtrp

```
3421 \newrobustcmd*{\@glsxtrp}[2]{%
3422   \ifcsdef{gls#1}%
3423   {%
3424     \@glsxtrp{gls#1}{#2}%
3425   }%
3426   {%
3427     \ifcsdef{glsxtr#1}%
3428     {%
3429       \@glsxtrp{glsxtr#1}{#2}%
3430     }%
3431     {%
3432       \PackageError{glossaries-extra}{‘#1’ not recognised by
3433         \string\glsxtrp}{}%
3434     }%
3435   }%
3436 }
```

\@Glsxtrp

```
3437 \newrobustcmd*{\@Glsxtrp}[2]{%
3438   \ifcsdef{Gls#1}%
3439   {%
3440     \@glsxtrp{Gls#1}{#2}%
3441   }%
3442   {%
3443     \ifcsdef{Glsxtr#1}%
3444     {%
3445       \@glsxtrp{Glsxtr#1}{#2}%
3446     }%
3447     {%
3448       \PackageError{glossaries-extra}{‘#1’ not recognised by
3449         \string\Glsxtrp}{}%
3450     }%
3451   }%
3452 }
```

\@GLSxtrp

```
3453 \newrobustcmd*{\@GLSxtrp}[2]{%
3454   \ifcsdef{GLS#1}%
3455   {%
3456     \@glsxtrp{GLS#1}{#2}%
3457   }%
3458   {%
3459     \ifcsdef{GLSxtr#1}%
3460     {%
3461       \@glsxtrp{GLSxtr#1}{#2}%
3462     }%
3463     {%
3464       \PackageError{glossaries-extra}{‘#1’ not recognised by
```

```

3465      \string\GLSxtrp}{}%
3466    }%
3467  }%
3468 }

```

\glsxtr@entry@p

```

3469 \newrobustcmd*{\glsxtr@headentry@p}[2]{%
3470   \glsifattribute{#1}{headuc}{true}%
3471   {%
3472     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
3473   }%
3474   {%
3475     \@gls@entry@field{#1}{#2}%
3476   }%
3477 }

```

\glsxtrp Not robust as it needs to expand somewhat.

```

3478 \ifdef\teorpdfstring
3479 {
3480   \newcommand{\glsxtrp}[2]{%
3481     \protect\NoCaseChange
3482     {%
3483       \protect\teorpdfstring
3484       {%
3485         \protect\glsxtrifinmark
3486         {%
3487           \ifcsdef{glsxtrhead#1}%
3488             {%
3489               {\protect\csuse{glsxtrhead#1}{#2}}%
3490             }%
3491             {%
3492               \glsxtr@headentry@p{#2}{#1}%
3493             }%
3494           }%
3495           {%
3496             \@glsxtrp{#1}{#2}%
3497           }%
3498         }%
3499         {%
3500           \protect\@gls@entry@field{#2}{#1}%
3501         }%
3502       }%
3503     }
3504 }
3505 {
3506   \newcommand{\glsxtrp}[2]{%
3507     \protect\NoCaseChange
3508     {%
3509       \protect\glsxtrifinmark

```

```

3510      {%
3511      \ifcsdef{glxtrhead#1}%
3512      {%
3513          {\protect\csuse{glxtrhead#1}}%
3514      }%
3515      {%
3516          \glxtr@headentry@p{#2}{#1}%
3517      }%
3518      }%
3519      {%
3520          \@glxtrp{#1}{#2}%
3521      }%
3522      }%
3523  }
3524 }

```

Provide short synonyms for the most common option.

`\glsp`

```

3525 \newcommand*{\glsp}{\glxtrp{short}}

```

`\glsp`

```

3526 \newcommand*{\glsp}{\glxtrp{text}}

```

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

```

3527 \ifdef\teorpdfstring
3528 {
3529   \newcommand{\Glsxtrp}[2]{%
3530     \protect\NoCaseChange
3531     {%
3532       \protect\teorpdfstring
3533       {%
3534         \protect\glxtrifinmark
3535         {%
3536           \ifcsdef{Glsxtrhead#1}%
3537           {%
3538             {\protect\csuse{Glsxtrhead#1}{#2}}%
3539           }%
3540           {%
3541             \protect\@Gls@entry@field{#2}{#1}%
3542           }%
3543         }%
3544         {%
3545           \@Glsxtrp{#1}{#2}%
3546         }%
3547       }%
3548       {%
3549         \protect\@Gls@entry@field{#2}{#1}%

```

```

3550     }%
3551 }%
3552 }
3553 }
3554 {
3555   \newcommand{\Glsxtrp}[2]{%
3556     \protect\NoCaseChange
3557     {%
3558       \protect\glsxtrifinmark
3559       {%
3560         \ifcsdef{Glsxtrhead#1}%
3561         {%
3562           {\protect\csuse{Glsxtrhead#1}}%
3563         }%
3564         {%
3565           \protect\@Gls@entry@field{#2}{#1}%
3566         }%
3567       }%
3568     }%
3569     \@Glsxtrp{#1}{#2}%
3570   }%
3571 }%
3572 }
3573 }

```

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

```

3574 \ifdef\texorpdfstring
3575 {
3576   \newcommand{\Glsxtrp}[2]{%
3577     \protect\NoCaseChange
3578     {%
3579       \protect\texorpdfstring
3580       {%
3581         \protect\glsxtrifinmark
3582         {%
3583           \ifcsdef{Glsxtr#1}%
3584           {%
3585             {\protect\Glsxtrshort[noindex,hyper=false]{#1}[]}%
3586           }%
3587           {%
3588             \protect\mfirstucMakeUppercase
3589             {%
3590               \protect\@gls@entry@field{#2}{#1}%
3591             }%
3592           }%
3593         }%
3594         {%
3595           \@Glsxtrp{#1}{#2}%
3596         }%

```

```

3597     }%
3598     {%
3599         \protect\@gls@entry@field{#2}{#1}%
3600     }%
3601 }%
3602 }
3603 }
3604 {
3605     \newcommand{\GLSxtrp}[2]{%
3606         \protect\NoCaseChange
3607         {%
3608             \protect\glsxtrifinmark
3609             {%
3610                 \ifcsdef{GLSxtr#1}%
3611                 {%
3612                     {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
3613                 }%
3614                 {%
3615                     \protect\mfirstucMakeUppercase
3616                     {%
3617                         \protect\@gls@entry@field{#2}{#1}%
3618                     }%
3619                 }%
3620             }%
3621             {%
3622                 \@GLSxtrp{#1}{#2}%
3623             }%
3624         }%
3625     }
3626 }

```

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

```

\@glsxtr@unset  Global unset.
3627 \newcommand*{\@glsxtr@unset}[1]{%

```

```

3628 \@@glsunset{#1}%
3629 \glsxtrpostunset{#1}%
3630 }%

\@glsunset Global unset.
3631 \let\@glsunset\@glsxtr@unset

glsxtrpostunset
3632 \newcommand*{\glsxtrpostunset}[1]{%

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

tUnsetBuffering
3633 \newcommand*{\GlsXtrStartUnsetBuffering}{%
3634 \ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
3635 }

tUnsetBuffering Unstarred version doesn't check for duplicates.
3636 \newcommand*{\@GlsXtrStartUnsetBuffering}{%
3637 \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3638 \def\@glsxtr@unset@buffer{}}%
3639 \let\@glsunset\@glsxtrbuffer@unset
3640 }

tUnsetBuffering Starred version checks for duplicates.
3641 \newcommand*{\s@GlsXtrStartUnsetBuffering}{%
3642 \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
3643 \def\@glsxtr@unset@buffer{}}%
3644 \let\@glsunset\@glsxtrbuffer@nodup@unset
3645 }

xtrbuffer@unset This must use a global change since \gls may have to be placed inside \mbox (for example,
with soul commands).
3646 \newcommand*{\@glsxtrbuffer@unset}[1]{%
3647 \listxadd\@glsxtr@unset@buffer{#1}%
3648 }

fer@nodup@unset Alternative version that avoids duplicates. One level of expansion is performed on the ar-
gument in case it's a control sequence containing the label. (Not using \xifinlist as the
added complexity might cause problems that the buffering is trying to overcome.)
3649 \newcommand*{\@glsxtrbuffer@nodup@unset}[1]{%
3650 \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}}%
3651 {\listxadd\@glsxtr@unset@buffer{#1}}%
3652 }

pUnsetBuffering
3653 \newcommand*{\GlsXtrStopUnsetBuffering}{%
3654 \ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
3655 }

```


pUnsetBuffering Unstarred form (global unset).

```
3656 \newcommand*{\@GlsXtrStopUnsetBuffering}{%
3657   \let\@glsunset\@glxtr@unset
3658   \forlistloop\@glsunset\@glxtr@unset@buffer
3659   \let\@glxtr@unset@buffer\@glxtr@org@unset@buffer
3660 }
```

pUnsetBuffering Starred form (local unset).

```
3661 \newcommand*{\s@GlsXtrStopUnsetBuffering}{%
3662   \forlistloop\@glslocalunset\@glxtr@unset@buffer
3663   \let\@glsunset\@glxtr@unset
3664 }
```

dUnsetBuffering Discards pending buffer and restores \glsunset.

```
3665 \newcommand*{\GlsXtrDiscardUnsetBuffering}{%
3666   \let\@glsunset\@glxtr@unset
3667   \let\@glxtr@unset@buffer\@glxtr@org@unset@buffer
3668 }
```

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

```
3669 \newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
3670   \forlistloop#1\@glxtr@unset@buffer
3671 }
```

\@glslocalunset Local unset.

```
3672 \renewcommand*{\@glslocalunset}[1]{%
3673   \@glslocalunset{#1}%
3674   \glxtrpostlocalunset{#1}%
3675 }%
```

rpostlocalunset

```
3676 \newcommand*{\glxtrpostlocalunset}[1]{}
```

\@glsreset Global reset.

```
3677 \renewcommand*{\@glsreset}[1]{%
3678   \@glsreset{#1}%
3679   \glxtrpostreset{#1}%
3680 }%
```

glxtrpostreset

```
3681 \newcommand*{\glxtrpostreset}[1]{}
```

\@glslocalreset Local reset.

```
3682 \renewcommand*{\@glslocalreset}[1]{%
3683   \@glslocalreset{#1}%
3684   \glxtrpostlocalreset{#1}%
3685 }%
```

trpostlocalreset

```
3686 \newcommand*{\glxtrpostlocalreset}[1]{}
```

sllocalreseteach Locally reset a list of entries.

```
3687 \newcommand*{\glsllocalreseteach}[1]{%
3688   \gl@ifnotmeasuring
3689   {%
3690     \@for\@gls@thislabel:=#1\do{%
3691       \gl@ifexists{\@gls@thislabel}%
3692       {%
3693         \glsllocalreset{\@gls@thislabel}%
3694       }%
3695     }%
3696   }%
3697 }
```

sllocalunseteach Locally unset a list of entries.

```
3698 \newcommand*{\glsllocalunseteach}[1]{%
3699   \gl@ifnotmeasuring
3700   {%
3701     \@for\@gls@thislabel:=#1\do{%
3702       \gl@ifexists{\@gls@thislabel}%
3703       {%
3704         \glsllocalunset{\@gls@thislabel}%
3705       }%
3706     }%
3707   }%
3708 }
```

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

```
3709 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

 Enable entry counting:

```
3710   \gl@enableentrycount
```

 Redefine \gls etc:

```
3711   \renewcommand*{\gls}{\cglsl}%
3712   \renewcommand*{\Gls}{\cGls}%
3713   \renewcommand*{\glspl}{\cglspl}%
3714   \renewcommand*{\Glspl}{\cGlspl}%
3715   \renewcommand*{\GLS}{\cGLS}%
3716   \renewcommand*{\GLSpl}{\cGLSpl}%
```

 Set the entrycount attribute:

```
3717   \@glxtr@setentrycountunsetattr{#1}{#2}%
```

 In case this command is used again:

```
3718   \let\GlsXtrEnableEntryCounting\@glxtr@setentrycountunsetattr
3719   \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
```

```

3720 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
3721   can't be used with \string\GlsXtrEnableEntryCounting}%
3722 {Use one or other but not both commands}}%
3723 }

```

ycountunsetattr

```

3724 \newcommand*{\@glxtr@setentrycountunsetattr}[2]{%
3725   \@for\@glxtr@cat:=#1\do
3726   {%
3727     \ifdefempty{\@glxtr@cat}{}%
3728     {%
3729       \glsssetcategoryattribute{\@glxtr@cat}{entrycount}{#2}%
3730     }%
3731   }%
3732 }

```

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

nableentrycount

```

3733 \renewcommand*{\glsenableentrycount}{%

```

Enable new fields:

```

3734 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%

```

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

```

3735 \renewcommand*{\gls@defdocnewglossaryentry}{%
3736   \renewcommand*{\newglossaryentry}[2]{%
3737     \PackageError{glossaries}{\string\newglossaryentry\space
3738       may only be used in the preamble when entry counting has
3739       been activated}{If you use \string\glsenableentrycount\space
3740       you must place all entry definitions in the preamble not in
3741       the document environment}%
3742   }%
3743 }%

```

New commands to access new fields:

```

3744 \newcommand*{\glsetentrycurrcount}[1]{%
3745   \ifcsundef{glo@\glsetoklabel{##1}@currcount}%
3746   {0}{\@gls@entry@field{##1}{currcount}}%
3747 }%
3748 \newcommand*{\glsetentryprevcount}[1]{%
3749   \ifcsundef{glo@\glsetoklabel{##1}@prevcount}%
3750   {0}{\@gls@entry@field{##1}{prevcount}}%
3751 }%

```

Adjust post unset and reset:

```

3752 \let\@glxtr@entrycount@org@unset\glxtrpostunset
3753 \renewcommand*{\glxtrpostunset}[1]{%
3754   \@glxtr@entrycount@org@unset{##1}%
3755   \@gls@increment@currcount{##1}%
3756 }%

```

```

3757 \let\@glxtr@entrycount@org@localunset\glxtrpostlocalunset
3758 \renewcommand*{\glxtrpostlocalunset}[1]{%
3759   \@glxtr@entrycount@org@localunset{##1}%
3760   \@gls@local@increment@currcount{##1}%
3761 }%
3762 \let\@glxtr@entrycount@org@reset\glxtrpostreset
3763 \renewcommand*{\glxtrpostreset}[1]{%
3764   \@glxtr@entrycount@org@reset{##1}%
3765   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
3766 }%
3767 \let\@glxtr@entrycount@org@localreset\glxtrpostlocalreset
3768 \renewcommand*{\glxtrpostlocalreset}[1]{%
3769   \@glxtr@entrycount@org@localreset{##1}%
3770   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
3771 }%

```

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

```

3772 \let\@cgls@\@cgls@
3773 \let\@cglspl@\@cglspl@

3774 \let\@cGls@\@cGls@
3775 \let\@cGlspl@\@cGlspl@
3776 \let\@cGLS@\@cGLS@
3777 \let\@cGLSpl@\@cGLSpl@

```

The rest is as the original definition.

```

3778 \AtEndDocument{\@gls@write@entrycounts}%
3779 \renewcommand*{\@gls@entry@count}[2]{%
3780   \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
3781 }%
3782 \let\glsenableentrycount\relax
3783 \renewcommand*{\glsenableentryunitcount}{%
3784   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
3785     can't be used with \string\glsenableentrycount}%
3786   {Use one or other but not both commands}%
3787 }%
3788 }

```

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

```

3789 \renewcommand*{\@gls@write@entrycounts}{%
3790   \immediate\write\@auxout
3791     {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
3792   \count@=0\relax
3793   \forallglsentries{\@glsentry}{%
3794     \gls@hasattribute{\@glsentry}{entrycount}%
3795     {%
3796       \ifglsused{\@glsentry}%
3797       {%

```

```

3798     \immediate\write\@auxout
3799     {\string\@gls@entry@count{\@glsentry}{\@glsentrycurrcount{\@glsentry}}}%
3800   }%
3801   {}%
3802   \advance\count@ by \@ne
3803   }%
3804   {}%
3805 }%
3806 \ifnum\count@=0
3807   \GlossariesExtraWarningNoLine{Entry counting has been enabled
3808   \MessageBreak with \string\glsenableentrycount\space but the
3809   \MessageBreak attribute ‘entrycount’ hasn’t
3810   \MessageBreak been assigned to any of the defined
3811   \MessageBreak entries}%
3812 \fi
3813 }

```

rifcounttrigger

<code>\glxtrifcounttrigger{<label>}{<trigger format>}{<normal>}</code>
--

```

3814 \newcommand*{\glxtrifcounttrigger}[3]{%
3815   \glshasattribute{#1}{entrycount}%
3816   {%
3817     \ifnum\glsentryprevcount{#1}>\glsgetattribute{#1}{entrycount}\relax
3818     #3%
3819   \else
3820     #2%
3821   \fi
3822   }%
3823   {#3}%
3824 }

```

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

\@@cgl@

```

3825 \def\@@cgl@#1#2[#3]{%
3826   \glxtrifcounttrigger{#2}%
3827   {%
3828     \cglformat{#2}{#3}%
3829     \glset{#2}%
3830   }%
3831   {%
3832     \@gls@{#1}{#2}[#3]%
3833   }%
3834 }%

```

\@@cglsp1@

```
3835 \def\@@cglsp1@#1#2[#3]{%
3836   \glxtrifcounttrigger{#2}%
3837   {%
3838     \cglsp1format{#2}{#3}%
3839     \glset{#2}%
3840   }%
3841   {%
3842     \@glsp1@{#1}{#2}[#3]%
3843   }%
3844 }%
```

\@@cGls@

```
3845 \def\@@cGls@#1#2[#3]{%
3846   \glxtrifcounttrigger{#2}%
3847   {%
3848     \cGlsformat{#2}{#3}%
3849     \glset{#2}%
3850   }%
3851   {%
3852     \@Gls@{#1}{#2}[#3]%
3853   }%
3854 }%
```

\@@cGlsp1@

```
3855 \def\@@cGlsp1@#1#2[#3]{%
3856   \glxtrifcounttrigger{#2}%
3857   {%
3858     \cGlsp1format{#2}{#3}%
3859     \glset{#2}%
3860   }%
3861   {%
3862     \@Glsp1@{#1}{#2}[#3]%
3863   }%
3864 }%
```

\@@cGLS@

```
3865 \def\@@cGLS@#1#2[#3]{%
3866   \glxtrifcounttrigger{#2}%
3867   {%
3868     \cGLSformat{#2}{#3}%
3869     \glset{#2}%
3870   }%
3871   {%
3872     \@GLS@{#1}{#2}[#3]%
3873   }%
3874 }%
```

\@@cGLSp1@

```

3875 \def\@cGLSpl@#1#2[#3]{%
3876   \glstrifcounttrigger{#2}%
3877   {%
3878     \cGLSplformat{#2}{#3}%
3879     \glset{#2}%
3880   }%
3881   {%
3882     \@GLSpl@{#1}{#2}[#3]%
3883   }%
3884 }%

```

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

```

\@cgl's@
3885 \def\@cgl's@#1#2[#3]{\@gl's@{#1}{#2}[#3]}

```

```

\@cGL's@
3886 \def\@cGL's@#1#2[#3]{\@GL's@{#1}{#2}[#3]}

```

```

\@cgl'spl@
3887 \def\@cgl'spl@#1#2[#3]{\@gl'spl@{#1}{#2}[#3]}

```

```

\@cGL'spl@
3888 \def\@cGL'spl@#1#2[#3]{\@GL'spl@{#1}{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

```

\cGLS
3889 \newrobustcmd*{\cGLS}{\@gl's@hyp@opt\@cGLS}

```

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

```

3890 \newcommand*{\@cGLS}[2][{}]{%
3891   \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[{}]}%
3892 }

```

```

\@cGLS@
3893 \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.

```

3894 \newcommand*{\cGLSformat}[2]{%
3895   \expandafter\mfirstucMakeUppercase\expandafter{\cgl'sformat{#1}{#2}}%
3896 }

```

```

\cGLSpl
3897 \newrobustcmd*{\cGLSpl}{\@gl's@hyp@opt\@cGLSpl}

```

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

```
3898 \newcommand*{\@cGLSp1}[2] [] {%
3899   \new@ifnextchar[{\@cGLSp1@{#1}{#2}}{\@cGLSp1@{#1}{#2} []}%
3900 }
```

\@cGLSp1@

```
3901 \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.

```
3902 \newcommand*{\cGLSplformat}[2] {%
3903   \expandafter\mfirstuc\MakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
3904 }
```

Modify the trigger formats to check for the regular attribute.

\cglformat

```
3905 \renewcommand*{\cglformat}[2] {%
3906   \glsifregular{#1}
3907   {\glsentryfirst{#1}}%
3908   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
3909 }
```

\cGlsformat

```
3910 \renewcommand*{\cGlsformat}[2] {%
3911   \glsifregular{#1}
3912   {\Glsentryfirst{#1}}%
3913   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
3914 }
```

\cglsplformat

```
3915 \renewcommand*{\cglsplformat}[2] {%
3916   \glsifregular{#1}
3917   {\glsentryfirstplural{#1}}%
3918   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}%#2%
3919 }
```

\cGlsplformat

```
3920 \renewcommand*{\cGlsplformat}[2] {%
3921   \glsifregular{#1}
3922   {\Glsentryfirstplural{#1}}%
3923   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}%#2%
3924 }
```

New code similar to above for unit counting.

defunitcounters

```
3925 \newcommand*{\@@newglossaryentry@defunitcounters}{%
```



```

3926 \edef\@glo@countunit{\csuse{@glxstr@categoryattr@@\@glo@category @unitcount}}%
3927 \ifdefvoid\@glo@countunit
3928 {}%
3929 {%
3930   \@glxstr@ifunitcounter{\@glo@countunit}%
3931   {}%
3932   {\expandafter\@glxstr@addunitcounter\expandafter{\@glo@countunit}}%
3933   }%
3934 }

```

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

```

3935 \newcommand*{\@glxstr@unitcountlist}{}

```

@addunitcounter

```

3936 \newcommand*{\@glxstr@addunitcounter}[1]{%
3937   \listadd{\@glxstr@unitcountlist}{#1}%
3938   \ifcsundef{glxstr@theunit@#1}
3939   {%
3940     \ifcsdef{theH#1}%
3941     {\csdef{glxstr@theunit@#1}{\csuse{theH#1}}}%
3942     {\csdef{glxstr@theunit@#1}{\csuse{the#1}}}%
3943   }%
3944   {}%
3945 }

```

r@ifunitcounter

```

3946 \newcommand*{\@glxstr@ifunitcounter}[3]{%
3947   \xifinlist{#1}{\@glxstr@unitcountlist}{#2}{#3}%
3948 }

```

urrentunitcount

```

3949 \newcommand*\@glxstr@currentunitcount[1]{%
3950   glo@\glsetoklabel{#1}@currunit@\glgetattribute{#1}{unitcount}.%
3951   \csuse{glxstr@theunit@\glgetattribute{#1}{unitcount}}%
3952 }

```

eviousunitcount

```

3953 \newcommand*\@glxstr@previousunitcount[1]{%
3954   glo@\glsetoklabel{#1}@prevunit@\glgetattribute{#1}{unitcount}.%
3955   \csuse{glxstr@theunit@\glgetattribute{#1}{unitcount}}%
3956 }

```

t@currunitcount

```

3957 \newcommand*{\@gl@increment@currunitcount}[1]{%
3958   \glshasattribute{#1}{unitcount}%
3959   {%
3960     \edef\@glxstr@csname{\@glxstr@currentunitcount{#1}}%
3961     \ifcsundef{\@glxstr@csname}%

```

```

3962   {%
3963     \csgdef{\@glsxtr@csname}{1}%
3964     \listcsxadd
3965       {glo@\glsdetoklabel{#1}@unitlist}%
3966       {\glsgetattribute{#1}{unitcount}.%
3967         \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3968     }%
3969   }%
3970   {%
3971     \csxdef{\@glsxtr@csname}%
3972       {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
3973     }%
3974   }%
3975   {}%
3976 }

```

t@currunitcount

```

3977 \newcommand*{\@gls@local@increment@currunitcount}[1]{%
3978   \gls@hasattribute{#1}{unitcount}%
3979   {%
3980     \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
3981     \ifcsundef{\@glsxtr@csname}%
3982     {%
3983       \csdef{\@glsxtr@csname}{1}%
3984       \listcseadd
3985         {glo@\glsdetoklabel{#1}@unitlist}%
3986         {\glsgetattribute{#1}{unitcount}.%
3987           \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}}%
3988       }%
3989     }%
3990     {%
3991       \csedef{\@glsxtr@csname}%
3992         {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
3993       }%
3994     }%
3995     {}%
3996 }

```

r@currunitcount

```

3997 \newcommand*{\@glsxtr@currunitcount}[2]{%
3998   \ifcsundef
3999     {glo@\glsdetoklabel{#1}@currunit@#2}%
4000     {0}%
4001     {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
4002 }%

```

r@prevunitcount

```

4003 \newcommand*{\@glsxtr@prevunitcount}[2]{%
4004   \ifcsundef

```

```

4005 {glo@\glsdetoklabel{#1}@prevunit@#2}%
4006 {0}%
4007 {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}%
4008 }%

```

entryunitcount

```

4009 \newcommand*\glsenableentryunitcount{%
  Enable new fields:
4010 \appto@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
4011 \renewcommand*\gls@defdocnewglossaryentry{%
4012 \renewcommand*\newglossaryentry[2]{%
4013 \PackageError{glossaries}{\string\newglossaryentry\space
4014 may only be used in the preamble when entry counting has
4015 been activated}{If you use \string\glsenableentryunitcount\space
4016 you must place all entry definitions in the preamble not in
4017 the document environment}%
4018 }%
4019 }%

```

New commands to access new fields:

```

4020 \newcommand*\glsentrycurrcount[1]{%
4021 \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
4022 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
4023 }%
4024 \newcommand*\glsentryprevcount[1]{%
4025 \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
4026 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
4027 }%

```

Access total count:

```

4028 \newcommand*\glsentryprevtotalcount[1]{%
4029 \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
4030 {0}%
4031 {%
4032 \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
4033 }%
4034 }%

```

Access max value:

```

4035 \newcommand*\glsentryprevmaxcount[1]{%
4036 \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
4037 {0}%
4038 {%
4039 \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
4040 }%
4041 }%

```

Adjust post unset and reset:

```

4042 \let@glsxtr@entryunitcount@org@unset\glsxtrpostunset

```

```

4043 \renewcommand*{\glxtrpostunset}[1]{%
4044   \@glxtr@entryunitcount@org@unset{##1}%
4045   \@glx@increment@currunitcount{##1}%
4046 }%
4047 \let\@glxtr@entryunitcount@org@localunset\glxtrpostlocalunset
4048 \renewcommand*{\glxtrpostlocalunset}[1]{%
4049   \@glxtr@entryunitcount@org@localunset{##1}%
4050   \@glx@local@increment@currunitcount{##1}%
4051 }%
4052 \let\@glxtr@entryunitcount@org@reset\glxtrpostreset
4053 \renewcommand*{\glxtrpostreset}[1]{%
4054   \glshasattribute{##1}{unitcount}%
4055   {%
4056     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
4057     \ifcsundef{\@glxtr@csname}%
4058     {}%
4059     {\csgdef{\@glxtr@csname}{0}}%
4060   }%
4061   {}%
4062 }%
4063 \let\@glxtr@entryunitcount@org@localreset\glxtrpostlocalreset
4064 \renewcommand*{\glxtrpostlocalreset}[1]{%
4065   \@glxtr@entryunitcount@org@localreset{##1}%
4066   \glshasattribute{##1}{unitcount}%
4067   {%
4068     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
4069     \ifcsundef{\@glxtr@csname}%
4070     {}%
4071     {\csdef{\@glxtr@csname}{0}}%
4072   }%
4073   {}%
4074 }%

```

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

```

4075 \let\@cgl@\@cgl@
4076 \let\@cgl@spl@\@cgl@spl@

4077 \let\@cGls@\@cGls@
4078 \let\@cGlspl@\@cGlspl@
4079 \let\@cGLS@\@cGLS@
4080 \let\@cGLSpl@\@cGLSpl@

```

Write information to the aux file.

```

4081 \AtEndDocument{\@glx@write@entryunitcounts}%
4082 \renewcommand*{\@glx@entry@unitcount}[3]{%
4083   \csgdef{glo@\glx@detoklabel{##1}@prevunit@##3}{##2}%
4084   \ifcsundef{glo@\glx@detoklabel{##1}@prevunittotal}%
4085   {\csgdef{glo@\glx@detoklabel{##1}@prevunittotal}{##2}}%
4086   {%

```

```

4087 \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
4088 \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
4089 }%
4090 \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
4091 {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
4092 {%
4093 \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
4094 \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
4095 \fi
4096 }%
4097 }%
4098 \let\glsenableentryunitcount\relax
4099 \renewcommand*{\glsenableentrycount}{%
4100 \PackageError{glossaries-extra}{\string\glsenableentrycount\space
4101 can't be used with \string\glsenableentryunitcount}%
4102 {Use one or other but not both commands}}%
4103 }%
4104 }
4105 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

4106 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

4107 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
4108 \immediate\write\@auxout
4109 {\string\@gls@entry@unitcount
4110 {\@glsentry}%
4111 {\@glsxtr@currunitcount{\@glsentry}{#1}%
4112 }%
4113 {#1}}%
4114 }

```

entryunitcounts

```

4115 \newcommand*{\@gls@write@entryunitcounts}{%
4116 \immediate\write\@auxout
4117 {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
4118 \count@=0\relax
4119 \forallglsentries{\@glsentry}{%
4120 \gls@hasattribute{\@glsentry}{unitcount}%
4121 {%
4122 \ifglsused{\@glsentry}%
4123 {%
4124 \forlistcsloop
4125 {\@gls@write@entryunitcounts@do}%
4126 {glo@\glsdetoklabel{\@glsentry}@unitlist}%
4127 }%
4128 {}%
4129 \advance\count@ by \@ne

```

```

4130 }%
4131 {}%
4132 }%
4133 \ifnum\count@=0
4134 \GlossariesExtraWarningNoLine{Entry counting has been enabled
4135 \MessageBreak with \string\glsenableentryunitcount\space but the
4136 \MessageBreak attribute ‘unitcount’ hasn’t
4137 \MessageBreak been assigned to any of the defined
4138 \MessageBreak entries}%
4139 \fi
4140 }

```

`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.

```

4141 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

4142 \glsenableentryunitcount

```

Redefine `\gls` etc:

```

4143 \renewcommand*{\gls}{\cglsl}%
4144 \renewcommand*{\Gls}{\cGls}%
4145 \renewcommand*{\glspl}{\cglspl}%
4146 \renewcommand*{\Glspl}{\cGlspl}%
4147 \renewcommand*{\GLS}{\cGLS}%
4148 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

4149 \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

4150 \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
4151 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
4152 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
4153 can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
4154 {Use one or other but not both commands}}%
4155 }

```

`tcountunsetattr`

```

4156 \newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
4157 \@for\@glsxtr@cat:=#1\do
4158 {%
4159 \ifdefempty{\@glsxtr@cat}{}%
4160 {%
4161 \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
4162 \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
4163 }%
4164 }%
4165 }

```

1.3.6 Acronym Modifications

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

`\SetGenericNewAcronym`

```
4166 \renewcommand*{\SetGenericNewAcronym}{%
    Make sure \RestoreAcronyms has been used.
4167 \ifdefequal\@addtoacronymlists\@glstr@org@addtoacronymlists
4168 {}%
4169 {%
4170 \GlossariesWarning{\string\SetGenericNewAcronym\space used
4171 without restoring base acronym functions with
4172 \string\RestoreAcronyms}%
4173 }%
4174 \let\@Gls@entryname\@Gls@acentryname

    Redefine \newacronym:
4175 \renewcommand{\newacronym}[4][{}]{%
4176 \ifdefempty{\@glsacronymlists}%
4177 {%
4178 \def\@glo@type{\acronymtype}%
4179 \setkeys{glossentry}{##1}%
4180 \DeclareAcronymList{\@glo@type}%
4181 }%
4182 {}%
4183 \glskeylisttok{##1}%
4184 \glslabeltok{##2}%
4185 \glsshorttok{##3}%
4186 \glslongtok{##4}%
4187 \newacronymhook
4188 \protected@edef\@do@newglossaryentry{%
4189 \noexpand\newglossaryentry{\the\glslabeltok}%
4190 {%
4191 type=\acronymtype,%
4192 name={\expandonce{\acronymentry{##2}}},%
4193 sort={\acronymssort{\the\glsshorttok}{\the\glslongtok}},%
4194 text={\the\glsshorttok},%
4195 short={\the\glsshorttok},%
4196 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4197 long={\the\glslongtok},%
4198 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4199 category=acronym,
4200 \GenericAcronymFields,%
4201 \the\glskeylisttok
```

```

4202     }%
4203     }%
4204     \@do@newglossaryentry
4205 }%
4206 \renewcommand*{\acrfullfmt}[3]{%
4207   \glslink{##1}{##2}{\genacrfullformat{##2}{##3}}}%
4208 \renewcommand*{\Acrfullfmt}[3]{%
4209   \glslink{##1}{##2}{\Genacrfullformat{##2}{##3}}}%
4210 \renewcommand*{\ACRfullfmt}[3]{%
4211   \glslink{##1}{##2}{%
4212     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}}%
4213 \renewcommand*{\acrfullplfmt}[3]{%
4214   \glslink{##1}{##2}{\genplacrfullformat{##2}{##3}}}%
4215 \renewcommand*{\Acrfullplfmt}[3]{%
4216   \glslink{##1}{##2}{\Genplacrfullformat{##2}{##3}}}%
4217 \renewcommand*{\ACRfullplfmt}[3]{%
4218   \glslink{##1}{##2}{%
4219     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}}%
4220 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}}%
4221 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}}%
4222 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}}%
4223 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}}%
4224 }

```

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:

```

4225 \let\@glxtr@org@setacronymstyle\setacronymstyle
4226 \let\@glxtr@org@newacronymstyle\newacronymstyle

```

Save the list of acronyms in case they are required.

`tr@acronymlists`

```

4227 \let\@glxtr@acronymlists\@glsacronymlists

```

`dtoacronymlists`

```

4228 \let\@glxtr@org@addtoacronymlists\@addtoacronymlists

```

`setacronymlists`

```

4229 \let\@glxtr@org@setacronymlists\SetAcronymLists

```

Need to provide a replacement for `\forallacronyms` since `\@glsacronymlists` isn't available.

`lsxtr@abbrlists`

```

4230 \newcommand{\@glxtr@abbrlists}{}

```

`breviationlists`

```

4231 \newcommand*{\forallabbreviationlists}[2]{%

```



```

4232 \for#1:=\@glxtr@abbrlists\do{\ifdefempty{#1}-{#2}}%
4233 }

```

bbreviationlist

```

4234 \newcommand*{\@glxtr@addabbreviationlist}[1]{%
4235 \edef\@glo@type{#1}%
4236 \ifdefempty\@glxtr@abbrlists
4237 {\let\@glxtr@abbrlists\@glo@type}%
4238 {%
4239 \ifdefequal\@glxtr@abbrlists\@glo@type
4240 }%
4241 {%
4242 \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@abbrlists}{}%
4243 {\eappto\@glxtr@abbrlists{,\@glo@type}}%
4244 }%
4245 }%
4246 }

```

\forallacronyms Modify to add warning.

```

4247 \renewcommand*{\forallacronyms}[2]{%
4248 \@glxtr@base@acrcmd\forallacronyms\forallabbreviationlists
4249 \@for#1:=\@glxacronymlists\do{\ifx#1\@empty\else#2\fi}%
4250 }

```

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

```

4251 \newcommand*{\MakeAcronymsAbbreviations}{%

```

Undo acronym display style:

```

4252 \@for\@gls@type:=\@glxacronymlists\do{%
4253 \csgdef{gls@\@gls@type @entryfmt}{\glsentryfmt}%
4254 }%

```

Save and clear acronym list.

```

4255 \let\@glxtr@acronymlists\@glxacronymlists
4256 \let\@glxacronymlists\@empty
4257 \let\@addtoacronymlists\@gobble
4258 \let\SetAcronymLists\@gobble

```

Warn if \acrshort etc are used.

```

4259 \let\@glxtr@base@acrcmd\@glxtr@base@acrcmd@warn

```

Redefine \newacronym to use same interface as \newabbreviation.

```

4260 \renewcommand*{\newacronym}[4][ ]{%
4261 \glxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
4262 }%
4263 \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
4264 \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
4265 \renewcommand*{\setacronymstyle}[1]{%

```

```

4266     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
4267     unavailable.
4268     Use \string\setabbreviationstyle[acronym]\space instead.
4269     The original acronym interface can be restored with
4270     \string\RestoreAcronyms}{}%
4271 }%
4272 \renewcommand*{\newacronymstyle}[1]{%
4273     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
4274     available unless you restore the original acronym interface with
4275     \string\RestoreAcronyms}%
4276     \@glxtr@org@newacronymstyle{##1}%
4277 }%
4278 }

```

Switch acronyms to abbreviations:

```

4279 \MakeAcronymsAbbreviations

```

RestoreAcronyms Restore acronyms to glossaries interface.

```

4280 \newcommand*{\RestoreAcronyms}{%
    Restore acronym list.
4281 \let\@glsacronymlists\@glxtr@acronymlists
4282 \let\@addtoacronymlists\@glxtr@org@addtoacronymlists
4283 \let\SetAcronymLists\@glxtr@org@setacronymlists

```

Suppress warnings if \acrshort etc are used.

```

4284 \let\@glxtr@base@acrcmd\@gobbletwo

```

Restore acronym display style:

```

4285 \@for\@gls@type:=\@glsacronymlists\do{%
4286     \SetDefaultAcronymDisplayStyle{\@gls@type}%
4287 }%

```

Switch to the generic acronym mechanism.

```

4288 \SetGenericNewAcronym
4289 \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
4290 \renewcommand{\acronymfont}[1]{##1}%
4291 \let\setacronymstyle\@glxtr@org@setacronymstyle
4292 \let\newacronymstyle\@glxtr@org@newacronymstyle

```

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

```

4293 \renewcommand*{\@gls@link@checkfirsthyper{%
4294     \ifglused{\glslabel}%
4295     {\let\glxtrifwasfirstuse\@secondoftwo}
4296     {\let\glxtrifwasfirstuse\@firstoftwo}%
4297     \@glxtr@org@checkfirsthyper
4298 }
4299 \glssetcategoryattribute{acronym}{regular}{false}%
4300 \setacronymstyle{long-short}%
4301 }

```

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

```
4302 \renewcommand*{\glsacspace}[1]{%
4303   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
4304   \ifdim\dimen@<\glsacspacemax~\else\space\fi
4305 }
```

`\glsacspacemax` Value used in the above.

```
4306 \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`

```
4307 \newcommand*{\@glxtr@reg@glosslist}{}
```

Save the original definition of `\makeglossaries`:

```
4308 \let\@glxtr@org@makeglossaries\makeglossaries
```

`omakeglossaries` glossaries v4.45 introduced `\@domakeglossaries` to provide a way of disabling `\makeglossaries`. If it hasn't been defined, define here to do its argument:

```
4309 \providecommand{\@domakeglossaries}[1]{#1}
```

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`

```
4310 \renewcommand*{\makeglossaries}[1][]{%
4311   \@domakeglossaries
4312   {%
4313     \@glxtr@if@record@only
4314     {%
4315       \PackageError{glossaries-extra}{\string\makeglossaries\space
4316         not permitted\MessageBreak with record=\@glxtr@record@setting\space
4317         package option}%
4318       {You may only use \string\makeglossaries\space with
4319         record=off or record=alsoindex options}%
4320     }%
4321     {%
4322       \ifblank{#1}%
4323       {\@glxtr@org@makeglossaries}%
4324       {%
4325         \ifx\@glxtr@record@setting\@glxtr@record@setting@alsoindex
```

```

4326      \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
4327      not permitted\MessageBreak with record=alsoindex package option}%
4328      {You may only use the hybrid \string\makeglossaries[...]\space with
4329      record=off option}%
4330      \else

```

\@gls@@automake@immediate was introduced to glossaries v4.42 so it may not be defined.

```

4331      \ifdef\@gls@@automake@immediate{\@gls@@automake@immediate}{}%
4332      \edef\@glsxtr@reg@glosslist{#1}%
4333      \ifundef{\glswrite}{\newwrite\glswrite}{}%
4334      \protected@write\@auxout{}{\string\providecommand
4335      \string\@glsorder[1]{}%
4336      \protected@write\@auxout{}{\string\providecommand
4337      \string\@istfilename[1]{}%
4338      \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
4339      \protected@write\@auxout{}{\string\@glsorder{\glsorder}}%
4340      \protected@write\@auxout{}{\string\glsxtr@makeglossaries{#1}}%
4341      \write\@auxout{\string\providecommand\string\@gls@reference[3]{}%

```

Iterate through each supplied glossary type and activate it.

```

4342      \@for\@glo@type:=#1\do{%
4343      \ifdefempty{\@glo@type}{\@makeglossary{\@glo@type}}%
4344      }%

```

New glossaries must be created before \makeglossaries:

```

4345      \renewcommand*\newglossary[4][]{%
4346      \PackageError{glossaries}{New glossaries
4347      must be created before \string\makeglossaries}{You need
4348      to move \string\makeglossaries\space after all your
4349      \string\newglossary\space commands}}%

```

Any subsequence instances of this command should have no effect.

```

4350      \let\@makeglossary\@gobble

```

Version 1.42 removed letting \makeglossary to \relax (no kernel redefs may be in effect).

```

4351      \renewcommand\makeglossaries[1][]{}%

```

Disable all commands that have no effect after \makeglossaries

```

4352      \@disable@onlypremakeg

```

Allow see key:

```

4353      \let\gls@checkseeallowed\relax

```

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

```

4354      \renewcommand*\@do@seeglossary[2]{%
4355      \glsdoifexists{##1}%
4356      {%
4357      \edef\@gls@label{\glsdetoklabel{##1}}%
4358      \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4359      \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4360      {\@glsxtr@org@doseeglossary{##1}{##2}}%

```

```

4361      {%
4362      \@@glsxtrwrglossmark
4363      \protected@write\@auxout{}\{
4364      \string\@gls@reference
4365      {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}}%
4366      }%
4367    }%
4368  }%
4369 }%

Adjust \@@do@@wrglossary
4370 \let\@glsxtr@@do@@wrglossary\@@do@@wrglossary
4371 \def\@@do@@wrglossary{%
4372   \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
4373   \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4374   {\@glsxtr@@do@@wrglossary}%
4375   {\@gls@noidx@glossary}%
4376 }%

Suppress warning about no \makeglossaries
4377 \let\warn@nomakeglossaries\relax
4378 \def\warn@noprintglossary{%
4379   \GlossariesWarningNoLine{No \string\printglossary\space
4380     or \string\printglossaries\space
4381     found.^^J(Remove \string\makeglossaries\space if you don't want
4382     any glossaries.)^^JThis document will not have a glossary}%
4383 }%

Only warn for glossaries not listed.
4384 \renewcommand{\@gls@noref@warn}[1]{%
4385   \edef\@gls@type{##1}%
4386   \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
4387   {%
4388     \GlossariesExtraWarning{Can't use
4389       \string\printnoidx@glossary[type={\@gls@type}]
4390       when '\@gls@type' is listed in the optional argument of
4391       \string\makeglossaries}%
4392   }%
4393   {%
4394     \GlossariesWarning{Empty glossary for
4395       \string\printnoidx@glossary[type={##1}].
4396       Rerun may be required (or you may have forgotten to use
4397       commands like \string\gls)}%
4398   }%
4399 }%

Adjust display number list to check for type:
4400 \renewcommand*\@glsdisplaynumberlist[1]{%
4401   \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4402   {\@glsxtr@idx@displaynumberlist{##1}}%
4403   {\@glsxtr@noidx@displaynumberlist{##1}}%

```

```

4404         }%
Adjust entry list:
4405         \renewcommand*{\glsentrynumberlist}[1]{%
4406             \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4407             {\@glsxtr@idx@entrynumberlist{##1}}%
4408             {\@glsxtr@noidx@entrynumberlist{##1}}%
4409         }%

```

Adjust number list loop

```

4410         \renewcommand*{\glsnumberlistloop}[2]{%
4411             \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4412             {%
4413                 \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
4414                     not available for glossary ‘##1’}{}%
4415             }%
4416             {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
4417         }%

```

Only sanitize sort for normal indexing glossaries.

```

4418         \renewcommand*{\glsprestandardsort}[3]{%
4419             \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
4420             {%
4421                 \glsdosanitizesort
4422             }%
4423             {%
4424                 \ifglssanitizesort
4425                 \@gls@noidx@sanitizesort
4426             \else
4427                 \@gls@noidx@nosanitizesort
4428             \fi
4429             }%
4430         }%

```

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

```

4431         \renewcommand*{\new@glossaryentry}[2]{%
4432             \PackageError{glossaries-extra}{Glossary entries must be defined
4433                 in the preamble\MessageBreak when you use the optional argument
4434                 of \string\makeglossaries}{Either move your definitions to the
4435                 preamble or don't use the optional argument of
4436                 \string\makeglossaries}%
4437         }%

```

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).

```

4438         \let\@glo@assign@sortkey\@glsxtr@mixed@assign@sortkey
4439         \renewcommand*{\@printgloss@setsort}{%

```

Need to extract just the type value.

```

4440         \expandafter\@glsxtr@gettype\expandafter,\@glsxtr@printglossopts,%

```

```

4441         type=\glsdefaulttype,\@end@glstr@gettype
4442     \def\@glo@sorttype{\@glo@default@sorttype}%
4443 }%

```

Check automake setting:

```

4444     \ifglautomake
4445     \renewcommand*{\@gls@doautomake}{%
4446         \@for\@gls@type:=\@glstr@reg@glosslist\do{%
4447             \ifdefempty{\@gls@type}}{\@gls@automake{\@gls@type}}%
4448         }%
4449     }%
4450 \fi

```

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

```

4451     \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
4452 \fi
4453 }%
4454 }%
4455 }%
4456 }

```

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.)

```

4457 \newcommand{\@glstr@orgprintglossary}[2]{%
4458     \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

4459     \def\glossarytitle{%
4460         \ifcsdef{\@glo@type\@glo@type @title}%
4461             {\csuse{\@glo@type\@glo@type @title}}%
4462             {\glossaryname}}%
4463     \def\glossarytoctitle{\glossarytitle}%
4464     \let\org@glossarytitle\glossarytitle
4465     \def\@glossarystyle{%
4466         \ifx\@glossary@default@style\relax
4467             \GlossariesWarning{No default glossary style provided \MessageBreak
4468                 for the glossary '\@glo@type'. \MessageBreak
4469                 Using deprecated fallback. \MessageBreak
4470                 To fix this set the style with \MessageBreak
4471                 \string\setglossarystyle\space or use the \MessageBreak
4472                 style key=value option}%
4473     \fi
4474 }%
4475 \def\gls@dotoc\toctitle{\glssettoc\toctitle{\@glo@type}}%
4476 \let\org@glossaryentrynumbers\glossaryentrynumbers

```

```

4477 \bgroup
4478   \@printgloss@setsort
4479   \setkeys{printgloss}{#1}%
4480   \ifx\glossarytitle\org@glossarytitle
4481   \else
4482     \cslet{@glotype@}@glo@type @title{\glossarytitle}%
4483   \fi
4484   \let\currentglossary@glo@type
4485   \let\org@glossaryentrynumbers\glossaryentrynumbers
4486   \let\glsnonextpages@glsnonextpages
4487   \let\glsnextpages@glsnextpages

4488   \glsextractivatenopost
4489   \gls@dotoc@title
4490   \@glossarystyle
4491   \let\gls@org@glossaryentryfield\glossentry
4492   \let\gls@org@glossarysubentryfield\subglossentry
4493   \renewcommand{\glossentry}[1]{%
4494     \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
4495     \gls@org@glossaryentryfield{##1}%
4496   }%
4497   \renewcommand{\subglossentry}[2]{%
4498     \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
4499     \gls@org@glossarysubentryfield{##1}{##2}%
4500   }%
4501   \@gls@preglossaryhook
4502   #2%
4503 \egroup
4504 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
4505 \global\let\warn@noprintglossary\relax
4506 }

```

`\glsextractivatenopost` Change `\nopostdesc` and `\glsxtrnopostpunc` to behave as they do in the glossary.

```

4507 \newcommand*{\glsextractivatenopost}{%
4508   \let\nopostdesc\@nopostdesc
4509   \let\glsxtrnopostpunc\@glsxtr@nopostpunc
4510 }

```

`\glsxtrnopostpunc`

```

4511 \newrobustcmd*{\glsxtrnopostpunc}{%

```

`\glsxtr@nopostpunc` Provide a command that works like `\nopostdesc` but only switches of the punctuation without suppressing the post-description hook.

```

4512 \newcommand{\@glsxtr@nopostpunc}{%
4513   \let\@glsxtr@org@postdescription\glspostdescription
4514   \ifglsnopostdot
4515     \renewcommand{\glspostdescription}{%
4516       \glsnopostdottrue
4517       \let\glspostdescription\@glsxtr@org@postdescription

```



```

4518 \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4519 \glxtrpostdescription
4520 \@glxtr@nopostpunc@postdesc}%
4521 \else
4522 \renewcommand{\glspostdescription}{%
4523 \let\glspostdescription\@glxtr@org@postdescription
4524 \let\glxtrrestorepostpunc\@glxtr@restore@postpunc
4525 \glxtrpostdescription
4526 \@glxtr@nopostpunc@postdesc}%
4527 \fi
4528 \glsnopostdotfalse
4529 }

```

stpostpunc@postdesc

```

4530 \newcommand*{\@glxtr@nopostpunc@postdesc}{%

```

restore@postpunc

```

4531 \newcommand*{\@glxtr@restore@postpunc}{%
4532 \def\@glxtr@nopostpunc@postdesc{%
4533 \@glxtr@org@postdescription
4534 \let\@glxtr@nopostpunc@postdesc\@empty
4535 \let\glxtrrestorepostpunc\@empty
4536 }%
4537 }

```

restorepostpunc Does nothing outside of glossary.

```

4538 \newcommand*{\glxtrrestorepostpunc}{%

```

\@printglossary Redefine.

```

4539 \renewcommand{\@printglossary}[2]{%
4540 \def\@glxtr@printglossopts{#1}%
4541 \@glxtr@orgprintglossary{#1}{#2}%
4542 }

```

Add a key that switches off the entry targets:

```

4543 \define@choicekey{printgloss}{target}
4544 [\@glxtr@printglossval\@glxtr@printglossnr]%
4545 {true,false}[true]%
4546 {%
4547 \ifcase\@glxtr@printglossnr
4548 \def\@glstarget{\glsdohypertarget}%
4549 \else
4550 \let\@glstarget\@secondoftwo
4551 \fi
4552 }

```

hypernameprefix

```

4553 \newcommand*{\@glxtrhypernameprefix}{%

```

New to v1.20:

```
4554 \define@key{printgloss}{targetnameprefix}{%
4555   \renewcommand{\@glxtrhypernameprefix}{#1}%
4556 }
```

```
4557 \define@key{printgloss}{prefix}{%
4558   \renewcommand{\glo@linkprefix}{#1}%
4559 }
```

```
4560 \define@key{printgloss}{label}{%
4561   \glxtrsetglossarylabel{#1}%
4562 }
```

etglossarylabel Set the label for subsequent glossaries. If the label is fixed (that is, doesn't change with each glossary) this will need to be scoped or changed again to prevent duplicate labels.

```
4563 \newcommand{\glxtrsetglossarylabel}[1]{%
4564   \renewcommand*{\@glossaryseclabel}{%
4565     \protected@edef\@currentlabelname{\glossarytoctitle}%
4566     \label{#1}%
4567   }%
4568 }
```

lsdohypertarget Redefine to insert \@glxtrhypernameprefix before the target name.

```
4569 \let\@glxtr@org@glsdohypertarget\glsdohypertarget
4570 \renewcommand{\glsdohypertarget}[2]{%
4571   \@glxtr@org@glsdohypertarget{\@glxtrhypernameprefix#1}{#2}%
4572 }
```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```
4573 \ifx\@glstarget\@glxtr@org@glsdohypertarget
4574   \def\@glstarget{\glsdohypertarget}%
4575 \fi
```

@makeglossaries For the benefit of makeglossaries

```
4576 \newcommand*{\glxtr@makeglossaries}[1]{}
```

@glxtr@gettype Get just the type.

```
4577 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
4578   \def\@glo@type{#2}%
4579 }
```

@assign@sortkey Assign the sort key.

```
4580 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
4581   \edef\@glo@type{\@glo@type}%
4582   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
4583   {%
4584     \@glo@no@assign@sortkey{#1}%
4585   }%
```

```

4586  {%
4587    \@@glo@assign@sortkey{#1}%
4588  }%
4589 }%

```

Display number list for the regular version:

splaynumberlist

```

4590 \let\@glxtr@idx@displaynumberlist\glsdisplaynumberlist

```

Display number list for the “noidx” version:

splaynumberlist

```

4591 \newcommand*{\@glxtr@noidx@displaynumberlist}[1]{%
4592   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4593   \ifdef\@gls@loclist
4594   {%
4595     \def\@gls@noidx@loclist@sep{%
4596       \def\@gls@noidx@loclist@sep{%
4597         \def\@gls@noidx@loclist@sep{%
4598           \glsnumlistsep
4599         }%
4600         \def\@gls@noidx@loclist@finalsep{\glsnumlistlastsep}%
4601       }%
4602     }%
4603     \def\@gls@noidx@loclist@finalsep{}%
4604     \def\@gls@noidx@loclist@prev{}%
4605     \forlistloop{\@glsnoidxdisplayloclisthandler}{\@gls@loclist}%
4606     \@gls@noidx@loclist@finalsep
4607     \@gls@noidx@loclist@prev
4608   }%
4609   {%
4610     \glxtrundeftag
4611     \glsdoifexists{#1}%
4612     {%
4613       \GlossariesWarning{Missing location list for ‘#1’. Either
4614         a rerun is required or you haven’t referenced the entry.}%
4615     }%
4616   }%
4617 }%
4618

```

And for the number list loop:

@numberlistloop

```

4619 \newcommand*{\@glxtr@noidx@numberlistloop}[3]{%
4620   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4621   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
4622   \let\@gls@org@glsseeformat\glsseeformat
4623   \let\@glsnoidxdisplayloc#2\relax

```

```

4624 \let\glsseeformat#3\relax
4625 \ifdef\@gls@loclist
4626 {%
4627   \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
4628 }%
4629 {%

4630   \glsxtrundeftag
4631   \glsdoifexists{#1}%
4632   {%
4633     \GlossariesWarning{Missing location list for ‘##1’. Either
4634       a rerun is required or you haven’t referenced the entry.}%
4635   }%
4636 }%
4637 \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
4638 \let\glsseeformat\@gls@org@glsseeformat
4639 }%

```

Same for entry number list.

entrynumberlist

```

4640 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
4641   \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
4642   \ifdef\@gls@loclist
4643   {%
4644     \glsnoidxloclist{\@gls@loclist}%
4645   }%
4646   {%

4647     \glsxtrundeftag
4648     \glsdoifexists{#1}%
4649     {%
4650       \GlossariesWarning{Missing location list for ‘#1’. Either
4651         a rerun is required or you haven’t referenced the entry.}%
4652     }%
4653   }%
4654 }%

```

entrynumberlist

```

4655 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\@glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

4656 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
4657   \protected@edef\@glsxtr@titlelabel{#1}%
4658   \ifvoid\@glsxtr@titlelabel
4659   {%
4660   {%
4661     \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
4662   }%
4663   \ifvoid\@glsxtr@titlelabel}%

```

```

4664 {%
4665   \DTLifint{#1}%
4666   {%
4667     \ifnum#1<256\relax
4668     \edef#2{\char#1\relax}%
4669     \else
4670     \edef#2{#1}%
4671     \fi
4672   }%
4673   {%
4674     \ifcsundef{#1groupname}%
4675     {\def#2{#1}}%
4676     {\letcs#2{#1groupname}}%
4677   }%
4678 }%
4679 {%
4680   \let#2\@glsxtr@titlelabel
4681 }%
4682 }

```

g@getgrouptitle Save original definition of \@gls@getgrouptitle

```

4683 \let\glsxtr@org@getgrouptitle\@gls@getgrouptitle

```

trgetgrouptitle Provide a user-level command to fetch the group title. The first argument is the group label. The second argument is a control sequence in which to store the title.

```

4684 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
4685   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4686   \@onelevel@sanitize\@glsxtr@titlelabel
4687   \ifcsdef{\@glsxtr@titlelabel}
4688   {\letcs{#2}{\@glsxtr@titlelabel}}%
4689   {\glsxtr@org@getgrouptitle{#1}{#2}}%
4690 }
4691 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

trsetgrouptitle Sets the title for the given group label.

```

4692 \newcommand{\glsxtrsetgrouptitle}[2]{%
4693   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4694   \@onelevel@sanitize\@glsxtr@titlelabel
4695   \protected@csxdef{\@glsxtr@titlelabel}{#2}%
4696 }

```

alsetgrouptitle As above put only locally defines the title.

```

4697 \newcommand{\glsxtrlocalsetgrouptitle}[2]{%
4698   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
4699   \@onelevel@sanitize\@glsxtr@titlelabel
4700   \protected@csedef{\@glsxtr@titlelabel}{#2}%
4701 }

```

`\glsnavigation` Redefine to use new user-level command.

```
4702 \renewcommand*{\glsnavigation}{%
4703   \def\@gls@between{}%
4704   \ifcsundef\@gls@hypergroup\list@{\@glo@type}%
4705   {%
4706     \def\@gls@list{}%
4707   }%
4708   {%
4709     \expandafter\let\expandafter\@gls@list
4710       \csname @gls@hypergroup\list@{\@glo@type}\endcsname
4711   }%
4712   \@for\@gls@tmp:=\@gls@list\do{%
4713     \@gls@between
4714     \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
4715     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
4716     \let\@gls@between\glshypernavsep
4717   }%
4718 }
```

`@noidx@glossary`

```
4719 \renewcommand*{\@print@noidx@glossary}{%
4720   \ifcsdef\@glsref@{\@glo@type}%
4721   {%
4722     \ifcsdef\@glo@sortmacro@{\@glo@sorttype}%
4723     {%
4724       \csuse{\@glo@sortmacro@{\@glo@sorttype}}{\@glo@type}%
4725     }%
4726     {%
4727       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
4728     }%
4729     \glossarysection[\glossarytoctitle]{\glossarytitle}%
4730     \glossarypreamble
```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```
4731   \def\@gls@currentlettergroup{}%
4732   \begin{theglossary}%
4733     \glossaryheader
4734     \glsresetentrylist
4735     \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}%
4736     \end{theglossary}%
4737     \glossarypostamble
4738   }%
4739   {%
```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```
4740     \glsxtrifemptyglossary{\@glo@type}%
4741     {}%
4742     {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
```

```

4743 \@gls@noref@warn{\@glo@type}%
4744 }%
4745 }

```

noidxdisplayloc Patch to check for range formations.

```

4746 \renewcommand*{\glsnoidxdisplayloc}[4]{%
4747 \setentrycounter[#1]{#2}%
4748 \@glsxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
4749 }

```

xtr@display@loc Patch to check for range formations.

```

4750 \def\@glsxtr@display@loc#1#2\end@glxtr@display@loc#3{%
4751 \ifx#1\relax
4752 \glsxtrdisplaystartloc{#2}{#3}%
4753 \else
4754 \ifx#1\relax
4755 \glsxtrdisplayendloc{#2}{#3}%
4756 \else
4757 \glsxtrdisplaysingleloc{#1#2}{#3}%
4758 \fi
4759 \fi
4760 }

```

isplaysingleloc Single location.

```

4761 \newcommand*{\glsxtrdisplaysingleloc}[2]{%
4762 \csuse{#1}{#2}%
4763 }

```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glsxtrlocrangefmt`.

displaystartloc Start of a location range.

```

4764 \newcommand*{\glsxtrdisplaystartloc}[2]{%
4765 \edef\glsxtrlocrangefmt{#1}%
4766 \ifx\glsxtrlocrangefmt\empty
4767 \def\glsxtrlocrangefmt{glsnumberformat}%
4768 \fi
4769 \expandafter\glsxtrdisplaysingleloc
4770 \expandafter{\glsxtrlocrangefmt}{#2}%
4771 }

```

trdisplayendloc End of a location range.

```

4772 \newcommand*{\glsxtrdisplayendloc}[2]{%
4773 \edef\@glsxtr@tmp{#1}%
4774 \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}%
4775 \ifx\glsxtrlocrangefmt\@glsxtr@tmp
4776 \else
4777 \GlossariesExtraWarning{Mismatched end location range
4778 (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%

```

```

4779 \fi
4780 \expandafter\glxtrdisplayendlochook\expandafter{\@glxtr@tmp}{#2}%
4781 \expandafter\glxtrdisplaysingleloc
4782 \expandafter{\glxtrlocrangefmt}{#2}%
4783 \def\glxtrlocrangefmt{}%
4784 }

```

splayendlochook Allow the user to hook into the end of range command.

```

4785 \newcommand*{\glxtrdisplayendlochook}[2]{}

```

sxtrlocrangefmt Current range format. Empty if not in a range.

```

4786 \newcommand*{\glxtrlocrangefmt}{}

```

setentrycounter Adjust \setentrycounter to save the original prefix.

```

4787 \renewcommand*{\setentrycounter}[2] [] {%
4788 \def\glxtrcounterprefix{#1}%
4789 \ifx\glxtrcounterprefix\@empty
4790 \def\@glo@counterprefix{.}%
4791 \else
4792 \def\@glo@counterprefix{.#1.}%
4793 \fi
4794 \def\glsetentrycounter{#2}%
4795 }

```

ls@removespaces Redefine to allow adjustments to location hyperlink.

```

4796 \def\@gls@removespaces#1 #2\@nil{%
4797 \toks@=\expandafter{\the\toks@#1}%
4798 \ifx\@#2\%
4799 \edef\@glo@tmp{\the\toks@}%
4800 \ifx\@glo@tmp\empty
4801 \else

```

Expand location (just in case \toks@ is needed for something else).

```

4802 \expandafter\glxtrlocationhyperlink\expandafter
4803 \glsetentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
4804 \fi
4805 \else
4806 \@gls@ReturnAfterFi{%
4807 \@gls@removespaces#2\@nil
4808 }%
4809 \fi
4810 }

```

cationhyperlink

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



```

4811 \newcommand*{\glxtrlocationhyperlink}[3]{%
4812   \ifdefvoid\glxtrsupplocationurl
4813   {%
4814     \GlsXtrInternalLocationHyperlink{#1}-{#2}-{#3}%
4815   }%
4816   {%
4817     \hyperref{\glxtrsupplocationurl}{-}{#1#2#3}-{#3}%
4818   }%
4819 }

```

supphypernumber

```

4820 \newcommand*{\glxtrsupphypernumber}[1]{%
4821   {%
4822     \glshasattribute{\glscurrententrylabel}{externallocation}%
4823     {%
4824       \def\glxtrsupplocationurl{%
4825         \glsgetaattribute{\glscurrententrylabel}{externallocation}}%
4826     }%
4827     {%
4828       \def\glxtrsupplocationurl{}%
4829     }%
4830     \glshypernumber{#1}%
4831   }%
4832 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

4833 \renewcommand{\@print@glossary}{%
4834   \makeatletter
4835   \@input{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4836   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
4837   {%
4838     {\glxtrNoGlossaryWarning{\@glo@type}}%
4839     \ifglsxindy
4840       \ifcsundef{@xdy@\@glo@type @language}%
4841       {%
4842         \edef\@do@auxoutstuff{%
4843           \noexpand\AtEndDocument{%
4844             \noexpand\immediate\noexpand\write\@auxout{%
4845               \string\providecommand\string\@xdylanguage[2]{}}%
4846             \noexpand\immediate\noexpand\write\@auxout{%
4847               \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
4848           }%
4849         }%
4850       }%
4851       {%
4852         \edef\@do@auxoutstuff{%

```

```

4853     \noexpand\AtEndDocument{%
4854         \noexpand\immediate\noexpand\write\@auxout{%
4855             \string\providecommand\string\@xdylanguage[2]{}}%
4856         \noexpand\immediate\noexpand\write\@auxout{%
4857             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
4858                 @language\endcsname}}%
4859     }%
4860 }%
4861 }%
4862 \@do@auxoutstuff
4863 \edef\@do@auxoutstuff{%
4864     \noexpand\AtEndDocument{%
4865         \noexpand\immediate\noexpand\write\@auxout{%
4866             \string\providecommand\string\@gls@codepage[2]{}}%
4867         \noexpand\immediate\noexpand\write\@auxout{%
4868             \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
4869     }%
4870 }%
4871 \@do@auxoutstuff
4872 \fi
4873 \renewcommand*{\@warn@nomakeglossaries}{%
4874     \GlossariesWarningNoLine{\string\makeglossaries\space
4875         hasn't been used,^^Jthe glossaries will not be updated}%
4876 }%
4877 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`oGlsWarningHead` Header message.

```

4878 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
4879     This document is incomplete. The external file associated with
4880     the glossary '#1' (which should be called \texttt{#2})
4881     hasn't been created.%
4882 }

```

`rningEmptyStart` No entries have been added to the glossary.

```

4883 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
4884     This has probably happened because there are no entries defined
4885     in this glossary.%
4886 }

```

`arningEmptyMain` The default “main” glossary is empty.

```

4887 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
4888     If you don't want this glossary,
4889     add \texttt{nomain} to your package option list when you load
4890     \texttt{glossaries-extra.sty}. For example:%
4891 }

```

`ingEmptyNotMain` A glossary that isn't the default “main” glossary is empty.

```

4892 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
4893 Did you forget to use \texttt{type=#1} when you defined your
4894 entries? If you tried to load entries into this glossary with
4895 \texttt{\string\loadglsentries} did you remember to use
4896 \texttt{[#1]} as the optional argument? If you did, check that
4897 the definitions in the file you loaded all had the type set
4898 to \texttt{\string\glsdefaulttype}.%
4899 }

```

WarningCheckFile Advisory message to check the file contents.

```

4900 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
4901 Check the contents of the file \texttt{#1}. If
4902 it's empty, that means you haven't indexed any of your entries in this
4903 glossary (using commands like \texttt{\string\gls} or
4904 \texttt{\string\glsadd}) so this list can't be generated.
4905 If the file isn't empty, the document build process hasn't been
4906 completed.%
4907 }

```

WarningAutoMake Message when automake option has been used.

```

4908 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
4909 You may need to rerun \LaTeX. If you already have, it may be that
4910 \TeX's shell escape doesn't allow you to run
4911 \ifglxindy xindy\else makeindex\fi. Check the
4912 transcript file \texttt{\jobname.log}. If the shell escape is
4913 disabled, try one of the following:
4914
4915 \begin{itemize}
4916 \item Run the external (Lua) application:
4917
4918 \texttt{makeglossaries-lite \string\jobname\string}
4919
4920 \item Run the external (Perl) application:
4921
4922 \texttt{makeglossaries \string\jobname\string}
4923 \end{itemize}
4924
4925 Then rerun \LaTeX\ on this document.
4926 \GlossariesExtraWarning{Rerun required to build the
4927 glossary '#1' or check \TeX's shell escape allows
4928 you to run \ifglxindy xindy\else makeindex\fi}%
4929 }

```

WarningMisMatch Mismatching \makenoidxglossaries.

```

4930 \newcommand{\GlsXtrNoGlsWarningMisMatch}{%
4931 You need to either replace \texttt{\string\makenoidxglossaries}
4932 with \texttt{\string\makeglossaries} or replace
4933 \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with

```

```

4934 \texttt{\string\printnoidxglossary}
4935 (or \texttt{\string\printnoidxglossaries}) and then rebuild
4936 this document.%.
4937 }

```

arningBuildInfo Build advice.

```

4938 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
4939   Try one of the following:
4940   \begin{itemize}
4941     \item Add \texttt{automake} to your package option list when you load
4942       \texttt{glossaries-extra.sty}. For example:
4943
4944       \texttt{\string\usepackage[automake]%
4945         \glsopenbrace glossaries-extra\glsclosebrace}
4946
4947     \item Run the external (Lua) application:
4948
4949     \texttt{makeglossaries-lite.lua \string"\jobname\string"}
4950
4951     \item Run the external (Perl) application:
4952
4953     \texttt{makeglossaries \string"\jobname\string"}
4954   \end{itemize}
4955
4956   Then rerun \LaTeX\ on this document.%.
4957 }

```

trRecordWarning Paragraph for record=only.

```

4958 \newcommand{\GlsXtrRecordWarning}[1]{%
4959   \texttt{\string\printglossary} doesn't work
4960   with the \texttt{record=only} package option
4961   use\par\texttt{\string\printunsrtglossary[type=#1]}\par
4962   instead (or change the package option).%.
4963 }

```

oGlsWarningTail Final paragraph.

```

4964 \newcommand{\GlsXtrNoGlsWarningTail}{%
4965   This message will be removed once the problem has been fixed.%.
4966 }

```

GlsWarningNoOut No out file created. Build advice.

```

4967 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
4968   The file \texttt{#1} doesn't exist. This most likely means you haven't used
4969   \texttt{\string\makeglossaries} or you have used
4970   \texttt{\string\nofiles}. If this is just a draft version of the
4971   document, you can suppress this message using the
4972   \texttt{nomissingglstext} package option.%.
4973 }

```

glossarywarning

```
4974 \newcommand*{\@glstr@defaultnoglossarywarning}[1]{%
4975 \glossarysection[\glossarytoctitle]{\glossarytitle}
4976 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
4977 \par
4978 \glstrifemptyglossary{#1}%
4979 {%
4980 \GlsXtrNoGlsWarningEmptyStart\space
4981 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
4982 \medskip
4983 \noindent\texttt{\string\usepackage[nomain\ifglacronym ,acronym\fi]%
4984 \glsoopenbrace glossaries-extra\glsclosebrace}
4985 \medskip
4986 }%
4987 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
4988 }%
4989 {%
4990 \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
4991 {%
4992 \GlsXtrNoGlsWarningCheckFile
4993 {\jobname.\csname @glotype@\@glo@type @out\endcsname}
4994
4995 \ifglautomake
4996
4997 \GlsXtrNoGlsWarningAutoMake{#1}
4998
4999 \else
5000
5001 \ifthenelse{\equal{#1}{main}}%
5002 {%
5003 \GlsXtrNoGlsWarningEmptyMain\par
5004 \medskip
5005 \noindent\texttt{\string\usepackage[nomain]%
5006 \glsoopenbrace glossaries-extra\glsclosebrace}
5007 \medskip
5008 }%
5009 {}%
5010
5011 \ifdequal\makeglossaries\@no@makeglossaries
5012 {%
5013 \GlsXtrNoGlsWarningMisMatch
5014 }%
5015 {%
5016 \GlsXtrNoGlsWarningBuildInfo
5017 }%
5018 \fi
5019 }%
5020 {%
5021 \GlsXtrNoGlsWarningNoOut
```

```

5022      {\jobname.\csname @glo@type @out\endcsname}%
5023    }%
5024  }%
5025  \par
5026  \GlsXtrNoGlsWarningTail
5027 }

```

glossarywarning Warn about using `\printglossary` with `record`

```

5028 \newcommand*{\@glxtr@record@noglossarywarning}[1]{%
5029   \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
5030   with record=only package option\MessageBreak(use
5031   \string\printunsrtglossary[type=#1])\MessageBreak
5032   instead (or change the package option)}%
5033   \glossarysection[\glossarytoctitle]{\glossarytitle}
5034   \GlsXtrRecordWarning{#1}
5035   \GlsXtrNoGlsWarningTail
5036 }

```

Provide some commands to accompany the `record` option for use with **bib2gls**.

ResourceOptions Default resource options.

```

5037 \newcommand*{\GlsXtrDefaultResourceOptions}{}

```

xtrresourcefile Since it's dangerous for an external application to create a file with a `.tex` extension, as from v1.11 this enforces a `.glstex` extension to avoid conflict.

```

5038 \newcommand*{\glxtrresourcefile}[2] [] {%

```

The `record` option can't be set after this command.

```

5039   \disable@keys{glossaries-extra.sty}{record}%
5040   \glxtr@writefields
5041   \ifdefempty\GlsXtrDefaultResourceOptions
5042   {%
5043     \protected@write\@auxout{\glxtrresourceinit}%
5044     {\string\glxtr@resource{#1}{#2}}%
5045   }%
5046   {%
5047     \protected@write\@auxout{\glxtrresourceinit}%
5048     {\string\glxtr@resource{\GlsXtrDefaultResourceOptions,#1}{#2}}%
5049   }%
5050   \let\@glxtr@org@see@noindex\@gls@see@noindex
5051   \let\@gls@see@noindex\relax
5052   \IfFileExists{#2.glstex}%
5053   {%

```

Can't scope `\@input` so save and restore the category code of `@` to allow for internal commands in the location list.

```

5054     \edef\@bibgls@restreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
5055     \makeatletter
5056     \@input{#2.glstex}%
5057     \@bibgls@restreat

```

If the record=nameref option has been set, check if this is supported by the installed version of bib2gls.

```

5058 \glstr@check@bibgls@nameref
5059 }%
5060 {%
5061 \GlossariesExtraWarning{No file ‘#2.glstex’}%
5062 }%
5063 \let\glstr@see@noindex\glstr@org@see@noindex
5064 }
5065 \onlypreamble\glstrresourcefile

```

@bibgls@nameref This will only warn after bib2gls has created the .glstex file, but there's way to check before.

```

5066 \newcommand{\glstr@check@bibgls@nameref}{%
5067 \ifx\glstr@record@setting\glstr@record@setting@nameref
5068 \ifdef\bibglshrefchar
5069 {}%
5070 {%
5071 \GlossariesExtraWarning{record=nameref requires at least
5072 version 1.8 of bib2gls}%
5073 }%
5074 \fi
5075 \let\glstr@check@bibgls@nameref\relax
5076 }

```

glstrresourceinit Code used during the protected write operation.

```

5077 \newcommand*{\glstrresourceinit}{}

```

glstrresourcecount

```

5078 \newcount\glstrresourcecount

```

glstrLoadResources Short cut that uses \glstrresourcefile with \jobname as the mandatory argument.

```

5079 \newcommand*{\GlsXtrLoadResources}[1][]{%
5080 \ifnum\glstrresourcecount=0\relax
5081 \glstrresourcefile[#1]{\jobname}%
5082 \else
5083 \glstrresourcefile[#1]{\jobname-\the\glstrresourcecount}%
5084 \fi
5085 \advance\glstrresourcecount by 1\relax
5086 }

```

glstr@resource

```

5087 \newcommand*{\glstr@resource}[2]{}

```

\glstr@fields

```

5088 \newcommand*{\glstr@fields}[1]{}

```

glstr@texencoding

```

5089 \newcommand*{\glstr@texencoding}[1]{}

```

\glxtr@langtag

```
5090 \newcommand*{\glxtr@langtag}[1]{}
```

@pluralsuffixes

```
5091 \newcommand*{\glxtr@pluralsuffixes}[4]{}
```

tr@shortcutsval

```
5092 \newcommand*{\glxtr@shortcutsval}[1]{}
```

sxtr@linkprefix

```
5093 \newcommand*{\glxtr@linkprefix}[1]{}
```

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```
5094 \newcommand*{\glxtr@writefields}{%
```

```
5095   \protected@write\@auxout{}%
```

```
5096     {\string\providecommand*{\string\glxtr@fields}[1]{}}%
```

```
5097   \protected@write\@auxout{}%
```

```
5098     {\string\providecommand*{\string\glxtr@resource}[2]{}}%
```

```
5099   \protected@write\@auxout{}%
```

```
5100     {\string\providecommand*{\string\glxtr@pluralsuffixes}[4]{}}%
```

```
5101   \protected@write\@auxout{}%
```

```
5102     {\string\providecommand*{\string\glxtr@shortcutsval}[1]{}}%
```

```
5103   \protected@write\@auxout{}%
```

```
5104     {\string\providecommand*{\string\glxtr@linkprefix}[1]{}}%
```

```
5105   \protected@write\@auxout{{\string\glxtr@fields{\@gls@keymap}}}%
```

```
5106   \protected@write\@auxout{}%
```

```
5107     {\string\providecommand*{\string\glxtr@record}[5]{}}%
```

```
5108   \ifx\@glxtr@record@setting\@glxtr@record@setting@nameref
```

```
5109     \protected@write\@auxout{}%
```

```
5110       {\string\providecommand*{\string\glxtr@record@nameref}[8]{}}%
```

```
5111   \fi
```

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.

```
5112   \ifdef\CurrentTrackedLanguageTag
```

```
5113     {%
```

```
5114       \protected@write\@auxout{{\%
```

```
5115         \string\glxtr@langtag{\CurrentTrackedLanguageTag}}%
```

```
5116     }%
```

```
5117   {%
```

```
5118     \protected@write\@auxout{{\string\glxtr@pluralsuffixes
```

```
5119       {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
```

```
5120       {\glxtrabbrvpluralsuffix}}%
```

```
5121   \ifdef\inputencodingname
```

```
5122     {%
```



```

5123 \protected@write\@auxout{}\string\glsxtr@texencoding{\inputencodingname}}%
5124 }%
5125 {%

```

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.)

```

5126 \@ifpackageloaded{fontspec}%
5127 {\protected@write\@auxout{}\string\glsxtr@texencoding{utf8}}}%
5128 {}%
5129 }%
5130 \protected@write\@auxout{}\string\glsxtr@shortcutsval{\@glsxtr@shortcutsval}}%

```

Prefix deferred until the beginning of the document in case it's redefined later in the preamble. This is picked up by bib2gls when the external option is used.

```

5131 \AtBeginDocument
5132 {\protected@write\@auxout{}\string\glsxtr@linkprefix{\glolinkprefix}}}%
5133 \let\glsxtr@writefields\relax

```

If the automake option is on, try running bib2gls if the aux file exists. This has to be done before the aux file is opened (so package options `automake=immediate` and `automake=true` are identical if just bib2gls is used). The double-quotes around `\jobname` have been removed (v1.19) since `\jobname` will include double-quotes if the file name has spaces.

```

5134 \ifglsautomake
5135 \IfFileExists{\jobname.aux}%
5136 {\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`.

```

5137 \ifx\@gls@doautomake\@gls@doautomake@err
5138 \let\@gls@doautomake\relax
5139 \fi
5140 \fi

```

Check if `order=letter` has been used by mistake (but not if `record=alsoindex` has been used).

```

5141 \@glsxtr@if@record@only
5142 {\ifdefstring{\glsorder}{letter}%
5143 {\GlossariesExtraWarningNoLine{Package option 'order=letter' isn't
5144 supported with 'record=\@glsxtr@record@setting'. Use 'break-at=none'
5145 resource option instead}}}%
5146 {}%
5147 }%
5148 {}%
5149 }

```

`do@automake@err`

```

5150 \newcommand*{\@gls@doautomake@err}{%
5151 \PackageError{glossaries}{You must use
5152 \string\makeglossaries\space with automake=true}
5153 {%
5154 Either remove the automake=true setting or

```

```

5155     add \string\makeglossaries\space to your document preamble.%
5156 }%
5157 }

```

Allow locations specific to a particular counter to be recorded.

`\glxtr@record`

```

5158 \newcommand*{\glxtr@record}[5]{}

```

`@record@nameref` Used with `record=nameref` to include current label information.

```

5159 \newcommand*{\glxtr@record@nameref}[8]{}

```

`r@counterrecord` Aux file command.

```

5160 \newcommand*{\glxtr@counterrecord}[3]{%
5161   \glxtrfieldlistgadd{#1}{record.#2}{#3}%
5162 }

```

`unterrecordhook` Hook used by `\@glxtr@dorecord`.

```

5163 \newcommand*{\@glxtr@counterrecordhook}{}

```

`trRecordCounter` Activate recording for a particular counter (identified in the argument).

```

5164 \newcommand*{\GlsXtrRecordCounter}[1]{%
5165   \@glxtr@recordcounter{#1}%
5166 }
5167 \@onlypreamble\GlsXtrRecordCounter

```

`docounterrecord`

```

5168 \newcommand*{\@glxtr@docounterrecord}[1]{%
5169   \protected@write\@auxout{}\string\glxtr@counterrecord
5170     {\@glx@label}{#1}{\csuse{the#1}}}%
5171 }

```

`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.

```

5172 \newcommand*{\glxtrglossentry}[1]{%
5173   \glxtrtitleorpdforheading
5174   {\@glxtrglossentry{#1}}%
5175   {\glsentryname{#1}}%
5176   {\glxtrheadname{#1}}%
5177 }

```

`lsxtrglossentry` Another test is needed in case `\@glxtrglossentry` has been written to the table of contents.

```

5178 \newrobustcmd*{\@glxtrglossentry}[1]{%
5179   \glxtrtitleorpdforheading
5180   {%
5181     \glsdoifexists{#1}%
5182     {%
5183       \begingroup
5184       \edef\glscurrententrylabel{\glsetoklabel{#1}}%
5185       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
5186       \ifglshasparent{#1}%
5187       {\GlsXtrStandaloneSubEntryItem{#1}}%
5188       {\glsetryitem{#1}}%
5189       \GlsXtrStandaloneEntryName{#1}%
5190     \endgroup
5191   }%
5192 }%
5193 {\glsetryname{#1}}%
5194 {\glxtrheadname{#1}}%
5195 }

```

StandaloneEntryName

```

5196 \newcommand*{\GlsXtrStandaloneEntryName}[1]{%
5197   \glstarget{#1}{\glossentryname{#1}}%
5198 }

```

StandaloneGlossaryType To make it easier to adjust the definition of `\currentglossary` within `\glxtrglossentry`, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```

5199 \newcommand{\GlsXtrStandaloneGlossaryType}{\glsetrytype{\glscurrententrylabel}}

```

StandaloneSubEntryItem Used for sub-entries in standalone format. The argument is the entry's label.

```

5200 \newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
5201   \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}{}}%
5202 }

```

glossentryother As `\glxtrglossentry` but uses a different field. First argument is code to use in the header. The second argument is the entry's label. The third argument is the internal field label. This needs to be expandable in case it occurs in a sectioning command so it can't have an optional argument.

```

5203 \newcommand*{\glxtrglossentryother}[3]{%
5204   \ifstrempy{#1}%
5205   {%
5206     \ifcsdef{glxtrhead#3}%
5207     {%
5208       \glxtrtitleorpdforheading
5209       {\@glxtrglossentryother{#2}{#3}{#1}}%
5210       {\@glsetry@field{#2}{#3}}%
5211       {\csuse{glxtrhead#3}{#2}}%
5212     }%

```

```

5213   {%
5214     \glstrtitleorpdforheading
5215     {\@glstrglossentryother{#2}{#3}{#1}}%
5216     {\@gls@entry@field{#2}{#3}}%
5217     {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
5218   }%
5219 }%
5220 {%
5221   \glstrtitleorpdforheading
5222   {\@glstrglossentryother{#2}{#3}{#1}}%
5223   {\@gls@entry@field{#2}{#3}}%
5224   {#1}%
5225 }%
5226 }

```

`glossentryother` As `\@glstrglossentry` but uses a different field.

```

5227 \newrobustcmd*{\@glstrglossentryother}[3]{%
5228   \glstrtitleorpdforheading
5229   {%
5230     \glsdoifexists{#1}%
5231     {%
5232       \begingroup
5233       \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
5234       \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
5235       \ifglshasparent{#1}%
5236       {\GlsXtrStandaloneSubEntryItem{#1}}%
5237       {\glsentryitem{#1}}%
5238       \GlsXtrStandaloneEntryOther{#1}%
5239     \endgroup
5240   }%
5241 }%
5242 {\@gls@entry@field{#1}{#2}}%
5243 {#3}%
5244 }

```

`aloneEntryOther`

```

5245 \newcommand*{\GlsXtrStandaloneEntryOther}[2]{%
5246   \glstarget{#1}{\glossentrynameother{#1}{#2}}%
5247 }

```

`ntunsrtglossary` Similar to `\printnoidxglossary` but it displays all entries defined for the given glossary without sorting.

```

5248 \newcommand*{\printunsrtglossary}{%
5249   \@ifstar\s@printunsrtglossary\@printunsrtglossary
5250 }

```

`ntunsrtglossary` Unstarred version.

```

5251 \newcommand*{\@printunsrtglossary}[1][{}]{%
5252   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%

```

5253 }

ntunsrtglossary Starred version.

```
5254 \newcommand*{\s@printunsrtglossary}[2][{}]{%
5255   \begingroup
5256     #2%
5257     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
5258   \endgroup
5259 }
```

unsrtglossaries Similar to \printnoidxglossaries but it displays all entries defined for the given glossary without sorting.

```
5260 \newcommand*{\printunsrtglossaries}{%
5261   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
5262 }
```

@unsrt@glossary

```
5263 \newcommand*{\@print@unsrt@glossary}{%
5264   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5265   \glossarypreamble
5266   check for empty list
5267   \glstrifemptyglossary{\@glo@type}%
5268   {%
5269     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
5270   }%
5271   \key@ifundefined{glossentry}{group}%
5272   {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
5273   {\let\@gls@getgrouptitle\@glstr@unsrt@getgrouptitle}%
5274   \def\@gls@currentlettergroup{}}%
```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```
5275 \def\@glstr@doglossary{%
5276   \begin{theglossary}%
5277   \glossaryheader
5278   \glsresetentrylist
5279 }%
5280 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
5281   :\expandafter=\cename{glolist@\@glo@type}\endcename\do{%
5282   \ifdefempty{\glscurrententrylabel}
5283   }%
5284   {%
```

Provide a hook (for example to measure width).

```
5285   \let\glstr@process\@firstofone
5286   \let\printunsrtglossaryskipentry
5287   \glstr@printunsrtglossaryskipentry
5288   \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%
```

Don't check group for child entries.

```

5289 \glxtr@process
5290 {%
5291 \ifglshasparent{\glscurrententrylabel}{}%
5292 {%
5293 \@glxtr@checkgroup\glscurrententrylabel
5294 \expandafter\appto\expandafter\@glxtr@doglossary\expandafter
5295 {\@glxtr@groupheading}%
5296 }%
5297 \eappto\@glxtr@doglossary{%
5298 \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
5299 }%
5300 }%
5301 }%
5302 \appto\@glxtr@doglossary{\end{theglossary}}%
5303 \printunsrtglossarypredoglossary
5304 \@glxtr@doglossary
5305 }%
5306 \glossarypostamble
5307 }

```

entryprocesshook

```

5308 \newcommand*{\printunsrtglossaryentryprocesshook}[1]{}

```

glossaryskipentry

```

5309 \newcommand*{\printunsrtglossaryskipentry}{%
5310 \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
5311 can only be used within \string\printunsrtglossaryentryprocesshook}{}%
5312 }

```

entryprocesshook

```

5313 \newcommand*{\@glxtr@printunsrtglossaryskipentry}{%
5314 \let\glxtr@process\@gobble
5315 }

```

rypredoglossary

```

5316 \newcommand*{\printunsrtglossarypredoglossary}{}

```

lossary@handler

```

5317 \newcommand{\@printunsrt@glossary@handler}[1]{%
5318 \xdef\glscurrententrylabel{#1}%
5319 \printunsrtglossaryhandler\glscurrententrylabel
5320 }

```

glossaryhandler

```

5321 \newcommand{\printunsrtglossaryhandler}[1]{%
5322 \glxtrunsrtdo{#1}%
5323 }

```

triflabelinlist

```
\glxstriflabelinlist{<label>}{<list>}{<true>}{<false>}
```

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of `\@gls@ifinlist` which ensures the label and list are fully expanded.

```
5324 \newrobustcmd*{\glxstriflabelinlist}[4]{%
5325   \protected@edef\@glxtr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
5326   \@glxtr@doiflabelinlist{#3}{#4}%
5327 }
```

srtglossaryunit

```
5328 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
5329   \s@printunsrtglossary[type=\gldefaulttype,#1]{%
5330     \printunsrtglossaryunitsetup{#2}%
5331   }%
5332 }
```

ossaryunitsetup

```
5333 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
5334   \renewcommand{\printunsrtglossaryhandler}[1]{%
5335     \glxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
5336     {\glxtrunsrtdo{##1}}%
5337   }%
5338   }%
```

Only the target names should have the prefixes adjusted as `\gls` etc need the original `\glolinkprefix`. The `\@gobble` part discards `\glolinkprefix`.

```
5339   \ifcsundef{theH#1}%
5340   {%
5341     \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
5342   }%
5343   {%
5344     \renewcommand*{\@glxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
5345   }%
5346   \renewcommand*{\glossarysection}[2][ ]{%
5347     \appto\glossarypostamble{\glspare\medskip\glspare}%
5348   }
```

srtglossaryunit

```
5349 \newcommand{\print@noop@unsrtglossaryunit}[2][ ]{%
5350   \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
5351     requires the record=only or record=alsoindex package option}{}%
5352 }
```

t@getgrouptitle

```
5353 \newrobustcmd*{\@glxtr@unsrt@getgrouptitle}[2]{%
```

```

5354 \protected@edef\@glxtr@titlelabel{glxtr@grouptitle@#1}%
5355 \@onelevel@sanitize\@glxtr@titlelabel
5356 \ifcsdef{\@glxtr@titlelabel}
5357 {\letcs{#2}{\@glxtr@titlelabel}}%
5358 {\def#2{#1}}%
5359 }

```

`\glxtrunsrtdo` Provide a user-level call to `\@glxtr@noidx@do` to make it easier to define a new handler.

```

5360 \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).

```

5361 \newcommand*{\glxtrgroupfield}{group}

```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while `\@glxtr@doglossary` is being constructed rather than in the handler.

`glxtr@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.

```

5362 \newcommand*{\@glxtr@checkgroup}[1]{%
5363   \def\@glxtr@groupheading{}%
5364   \key@ifundefined{glossentry}{group}%
5365   {%
5366     \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
5367     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
5368   }%
5369   {%
5370     \protected@edef\@glo@thislettergrp{%
5371       \csuse{glo@\glsdetoklabel{#1}@glxtrgroupfield}}%
5372     }%
5373     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5374     {}%
5375     {%
5376       \ifdefempty{\@gls@currentlettergroup}{}%
5377       {\def\@glxtr@groupheading{\glsgroupskip}}%
5378       \eappto\@glxtr@groupheading{%
5379         \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
5380       }%
5381     }%
5382     \let\@gls@currentlettergroup\@glo@thislettergrp
5383 }

```

`trLocationField` Stores the internal name of the location field.

```

5384 \newcommand*{\GlsXtrLocationField}{location}

```


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.

```

5385 \newcommand{\@glsxtr@noidx@do}[1]{%
5386   \ifglstryexists{#1}%
5387   {%
5388     \global\letcs{\@gls@loclist}{glo@glstetoklabel{#1}@loclist}%
5389     \global\letcs{\@gls@location}{glo@glstetoklabel{#1}@GlsXtrLocationField}%
5390     \ifglshasparent{#1}%
5391     {%
5392       \gls@level=\csuse{glo@glstetoklabel{#1}@level}\relax
5393       \ifdefvoid{\@gls@location}%
5394       {%
5395         \ifdefvoid{\@gls@loclist}%
5396         {%
5397           \subglossentry{\gls@level}{#1}{}%
5398         }%
5399         {%
5400           \subglossentry{\gls@level}{#1}%
5401           {%
5402             \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5403           }%
5404         }%
5405       }%
5406     }%
5407     \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
5408   }%
5409 }%
5410 {%
5411   \ifdefvoid{\@gls@location}%
5412   {%
5413     \ifdefvoid{\@gls@loclist}
5414     {%
5415       \glossentry{#1}{}%
5416     }%
5417     {%
5418       \glossentry{#1}%
5419       {%
5420         \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
5421       }%
5422     }%
5423   }%
5424   {%
5425     \glossentry{#1}%
5426     {%
5427       \glossaryentrynumbers{\@gls@location}%
5428     }%
5429   }%
5430 }%

```

```

5431 }%
5432 {}%
5433 }

```

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`

```

5434 \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.)

`r@providenewgls`

```

5435 \newcommand*\@glsxtr@providenewgls{%
5436   \protected@write\@auxout{}\string\providecommand{\string\@glsxtr@newglslike}[2]{}%
5437   \let\@glsxtr@providenewgls\relax
5438 }

```

`identifyglslike` Identify the command given in the second argument for the benefit of **bib2gls**.

```

5439 \newcommand\@glsxtridentifyglslike[2]{%
5440   \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
5441   }%
5442   {%
5443     \@glsxtr@providenewgls
5444     \protected@write\@auxout{}\string\@glsxtr@newglslike{#1}\string#2}%
5445   }%
5446 }

```

`\@glsxtrnewgls`

`\glsxtrnewgls[<options>]{<prefix>}{<cs>}{<inner cs name>}`

```

5447 \newcommand*\@glsxtrnewgls[4]{%
5448   \ifdef{#3}%
5449   {%
5450     \PackageError{glossaries-extra}{Command \string#3\space already
5451 defined}{}%
5452   }%
5453   {%

```

Write information to the aux file for bib2gls.

```

5454     \glsxtridentifyglslike{#2}{#3}%
5455     \ifcsdef{@#4like@#2}%
5456     {%

```

```

5457     \advance\@glsxtrnewgls@inner by \@ne
5458     \def\@glsxtrnewgls@innercsname{#@4like\number\@glsxtrnewgls@inner @#2}%
5459 }%
5460 {\def\@glsxtrnewgls@innercsname{#@4like@#2}}%
5461 \expandafter\newrobustcmd\expandafter*\expandafter
5462   #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
5463 \ifstrempy{#1}%
5464 {%
5465   \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][ ]{%
5466     \new@ifnextchar[%
5467       {\csname @#4@\endcsname{##1}{#2##2}}%
5468       {\csname @#4@\endcsname{##1}{#2##2}[ ]}%
5469     }%
5470   }%
5471 {%
5472   \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][ ]{%
5473     \new@ifnextchar[%
5474       {\csname @#4@\endcsname{#1,##1}{#2##2}}%
5475       {\csname @#4@\endcsname{#1,##1}{#2##2}[ ]}%
5476     }%
5477   }%
5478 }%
5479 }

```

\glsxtrnewgls

`\glsxtrnewgls[<options>]{<prefix>}{<cs>}`

The first argument prepends to the options and the second argument is the prefix.

```

5480 \newrobustcmd*{\glsxtrnewgls}[3][ ]{%
5481   \@glsxtrnewgls{#1}{#2}{#3}{gls}%
5482 }

```

glsxtrnewglslike Provide a way to conveniently define commands that behave like \gls, \glspl, \Gls and \Glspl with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

5483 \newrobustcmd*{\glsxtrnewglslike}[6][ ]{%
5484   \@glsxtrnewgls{#1}{#2}{#3}{gls}%
5485   \@glsxtrnewgls{#1}{#2}{#4}{glspl}%
5486   \@glsxtrnewgls{#1}{#2}{#5}{Gls}%
5487   \@glsxtrnewgls{#1}{#2}{#6}{Glspl}%
5488 }

```

glsxtrnewGLSlike Provide a way to conveniently define commands that behave like \GLS, \GLSpl with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```

5489 \newrobustcmd*{\glsxtrnewGLSlike}[4][ ]{%
5490   \@glsxtrnewgls{#1}{#2}{#3}{GLS}%

```

```

5491 \@glxstrnewgls{#1}{#2}{#4}{GLSpl}%
5492 }

```

`\glxstrnewrgls` As `\glxstrnewgls` but for `\rgls`.

```

5493 \newrobustcmd*{\glxstrnewrgls}[3][]{%
5494 \@glxstrnewgls{#1}{#2}{#3}{rgls}%
5495 }

```

`\glxstrnewrglslike` As `\glxstrnewglslike` but for `\rgls` etc.

```

5496 \newrobustcmd*{\glxstrnewrglslike}[6][]{%
5497 \@glxstrnewgls{#1}{#2}{#3}{rgls}%
5498 \@glxstrnewgls{#1}{#2}{#4}{rglspl}%
5499 \@glxstrnewgls{#1}{#2}{#5}{rGls}%
5500 \@glxstrnewgls{#1}{#2}{#6}{rGlspl}%
5501 }

```

`\glxstrnewGLSlike` As `\glxstrnewGLSlike` but for `\rGLS` etc.

```

5502 \newrobustcmd*{\glxstrnewGLSlike}[4][]{%
5503 \@glxstrnewgls{#1}{#2}{#3}{rGLS}%
5504 \@glxstrnewgls{#1}{#2}{#4}{rGLSpl}%
5505 }

```

Provide easy access to record count fields.

`\totalRecordCount` Access total record count. This is designed to be expandable. The argument is the label.

```

5506 \newcommand*{\GlsXtrTotalRecordCount}[1]{%
5507 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount}%
5508 {\csname glo@\glsdetoklabel{#1}@recordcount\endcsname}%
5509 {0}%
5510 }

```

`\sXtrRecordCount` Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```

5511 \newcommand*{\GlsXtrRecordCount}[2]{%
5512 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2}%
5513 {\csname glo@\glsdetoklabel{#1}@recordcount.#2\endcsname}%
5514 {0}%
5515 }

```

`\locationRecordCount` 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.

```

5516 \newcommand*{\GlsXtrLocationRecordCount}[3]{%
5517 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}}%
5518 {\csname glo@\glsdetoklabel{#1}@recordcount.#2.\glxstrdetoklocation{#3}\endcsname}%
5519 {0}%
5520 }

```

trdetoklocation

```
5521 \newcommand*{\glxtrdetoklocation}[1]{#1}
```

ablerecordcount

```
5522 \newcommand*{\glxtrenablerecordcount}{%
5523   \renewcommand*{\gls}{\rgls}%
5524   \renewcommand*{\Gls}{\rGls}%
5525   \renewcommand*{\glspl}{\rglspl}%
5526   \renewcommand*{\Glspl}{\rGlspl}%
5527   \renewcommand*{\GLS}{\rGLS}%
5528   \renewcommand*{\GLSpl}{\rGLSpl}%
5529 }
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
5530 \newcommand*{\glxtrrecordtriggervalue}[1]{%
5531   \GlsXtrTotalRecordCount{#1}%
5532 }
```

dCountAttribute

```
5533 \newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
5534   \@for\@glxtr@cat:=#1\do
5535   {%
5536     \ifdefempty{\@glxtr@cat}{}%
5537     {%
5538       \glssetcategoryattribute{\@glxtr@cat}{recordcount}{#2}%
5539     }%
5540   }%
5541 }
```

ifrecordtrigger

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

```
5542 \newcommand*{\glxtrifrecordtrigger}[3]{%
5543   \glshasattribute{#1}{recordcount}%
5544   {%
5545     \ifnum\glxtrrecordtriggervalue{#1}>\glsgetattribute{#1}{recordcount}\relax
5546     #3%
5547   \else
5548     #2%
5549   \fi
5550 }%
5551 {#3}%
5552 }
```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```

5553 \newcommand*{\@glstr@rglstrigger@record}[3]{%
5554   \edef\glslabel{\glsdetoklabel{#2}}%
5555   \let\@gls@link@label\glslabel
5556   \def\@glstr@thevalue{%
5557     \def\@glstr@theHvalue{\@glstr@thevalue}%
5558     \def\@glsnumberformat{glstriggerrecordformat}%
5559     \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
5560     \edef\glstype{\csname glo@\glslabel @type\endcsname}%
5561     \def\@glstr@thevalue{%
5562       \def\@glstr@theHvalue{\@glstr@thevalue}%
5563       \glstr@trinitwrgloss
5564       \glslinkpresetkeys
5565       \setkeys{glslink}{#1}%
5566       \glslinkpostsetkeys
5567       \ifdefempty{\@glstr@thevalue}%
5568         {%
5569           \@gls@saveentrycounter
5570         }%
5571         {%
5572           \let\theglsentrycounter\@glstr@thevalue
5573           \def\theHglentrycounter{\@glstr@theHvalue}%
5574         }%
5575         \ifglstr@trinitwrglossbefore
5576           \@do@wrglossary{#2}%
5577         \fi
5578         #3%
5579         \ifglstr@trinitwrglossbefore
5580         \else
5581           \@do@wrglossary{#2}%
5582         \fi
5583         \ifKV@glslink@local
5584           \glslocalunset{#2}%
5585         \else
5586           \glsunset{#2}%
5587         \fi
5588 }

```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```

5589 \newcommand*{\glstriggerrecordformat}[1]{%

```

\rgls

```

5590 \newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}

```

\@rgls

```

5591 \newcommand*{\@rgls}[2][ ]{%
5592   \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5593 }

```

```

\@rgls@
5594 \def\@rgls@#1#2[#3]{%
5595   \glxtrifrecordtrigger{#2}%
5596   {%
5597     \@glxtr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5598   }%
5599   {%
5600     \@rgls@{#1}{#2}[#3]%
5601   }%
5602 }%

\rglspl
5603 \newrobustcmd*{\rglspl}{\@rgls@hyp@opt\@rglspl}

\@rglspl
5604 \newcommand*{\@rglspl}[2][{}]{%
5605   \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
5606 }

\@rglspl@
5607 \def\@rglspl@#1#2[#3]{%
5608   \glxtrifrecordtrigger{#2}%
5609   {%
5610     \@glxtr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
5611   }%
5612   {%
5613     \@rglspl@{#1}{#2}[#3]%
5614   }%
5615 }%

\@rgls
5616 \newrobustcmd*{\@rgls}{\@rgls@hyp@opt\@rgls}

\@rgls
5617 \newcommand*{\@rgls}[2][{}]{%
5618   \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
5619 }

\@rgls@
5620 \def\@rgls@#1#2[#3]{%
5621   \glxtrifrecordtrigger{#2}%
5622   {%
5623     \@glxtr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
5624   }%
5625   {%
5626     \@rgls@{#1}{#2}[#3]%
5627   }%
5628 }%

```

```

\rGlspl
5629 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\rGlspl}

\@rGlspl
5630 \newcommand*{\@rGlspl}[2] [] {%
5631   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2} [] }%
5632 }

\@rGlspl@
5633 \def\@rGlspl@#1#2[#3]{%
5634   \glsxtrifrecordtrigger{#2}%
5635   {%
5636     \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5637   }%
5638   {%
5639     \@Glspl@{#1}{#2}[#3]%
5640   }%
5641 }%

\rGLS
5642 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\rGLS}

\@rGLS
5643 \newcommand*{\@rGLS}[2] [] {%
5644   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2} [] }%
5645 }

\@rGLS@
5646 \def\@rGLS@#1#2[#3]{%
5647   \glsxtrifrecordtrigger{#2}%
5648   {%
5649     \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%
5650   }%
5651   {%
5652     \@GLS@{#1}{#2}[#3]%
5653   }%
5654 }%

\rGLSpl
5655 \newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\rGLSpl}

\@rGLSpl
5656 \newcommand*{\@rGLSpl}[2] [] {%
5657   \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2} [] }%
5658 }

```



```

\@rGLSpl@
5659 \def \@rGLSpl@#1#2[#3]{%
5660   \glxtrifrecordtrigger{#2}%
5661   {%
5662     \@glxtr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
5663   }%
5664   {%
5665     \@GLSpl@{#1}{#2}[#3]%
5666   }%
5667 }%

\rglsformat
5668 \newcommand*{\rglsformat}[2]{%
5669   \glsifregular{#1}
5670   {\glsentryfirst{#1}}%
5671   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
5672 }

\rglsplformat
5673 \newcommand*{\rglsplformat}[2]{%
5674   \glsifregular{#1}
5675   {\glsentryfirstplural{#1}}%
5676   {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}%#2%
5677 }

\rGlsformat
5678 \newcommand*{\rGlsformat}[2]{%
5679   \glsifregular{#1}
5680   {\Glsentryfirst{#1}}%
5681   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
5682 }

\rGlsplformat
5683 \newcommand*{\rGlsplformat}[2]{%
5684   \glsifregular{#1}
5685   {\Glsentryfirstplural{#1}}%
5686   {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}%#2%
5687 }

\rGLSformat
5688 \newcommand*{\rGLSformat}[2]{%
5689   \expandafter\mfirstucMakeUppercase\expandafter{\rglsformat{#1}{#2}}%
5690 }

\rGLSplformat
5691 \newcommand*{\rGLSplformat}[2]{%
5692   \expandafter\mfirstucMakeUppercase\expandafter{\rglsplformat{#1}{#2}}%
5693 }

```

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`.

```
5694 \newcommand{\@glsxtr@do@inc@linkcount}{%
```

Does this entry have the linkcount attribute set?

```
5695 \glsifattribute{\glslabel}{linkcount}{true}%
```

```
5696 {%
```

Does the counter exist?

```
5697 \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
```

```
5698 {%
```

Counter doesn’t exist, so define it.

```
5699 \newcounter{glsxtr@linkcount@\glslabel}%
```

If linkcountmaster is set, add to counter reset.

```
5700 \glshasattribute{\glslabel}{linkcountmaster}%
```

```
5701 {%
```

Need to ensure values are fully expanded.

```
5702 \begingroup
```

```
5703 \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@\glslabel}%
```

```
5704 {\glsgetattribute{\glslabel}{linkcountmaster}}}%
```

```
5705 \x
```

```
5706 }%
```

```
5707 {}%
```

```
5708 }%
```

Increment counter:

```
5709 \glsxtrinlinkcounter{glsxtr@linkcount@\glslabel}%
```

```
5710 }%
```

```
5711 {}%
```

```
5712 }
```

`\rinc@linkcounter` May be redefined to use `\refstepcounter` if required.

```
5713 \newcommand*{\glsxtrinlinkcounter}[1]{\stepcounter{#1}}
```

`\linkCounterValue` Expands to the associated link counter register or 0 if not defined.

```
5714 \newcommand*{\GlsXtrLinkCounterValue}[1]{%
```

```
5715 \ifcsundef{c@glsxtr@linkcount@#1}{0}{\csname c@glsxtr@linkcount@#1\endcsname}%
```

```
5716 }
```

`rTheLinkCounter` Expands to the display value of the associated link counter or 0 if not defined.

```
5717 \newcommand*\GlsXtrTheLinkCounter}[1]{%
5718 \ifcsundef{theglsxtr@linkcount@#1}{0}%
5719 {\csname theglxtr@linkcount@#1\endcsname}%
5720 }
```

`fLinkCounterDef` Tests if the counter has been defined

```
5721 \newcommand*\GlsXtrIfLinkCounterDef}[3]{%
5722 \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
5723 }
```

`LinkCounterName` Expands to the associated link counter name. (No check for existence.)

```
5724 \newcommand*\GlsXtrLinkCounterName}[1]{glxtr@linkcount@#1}
```

`bleLinkCounting`

`\GlsXtrEnableLinkCounting[<master counter>]{<categories>}`

Enable link counting for the given categories.

```
5725 \newcommand*\GlsXtrEnableLinkCounting}[2][{}]{%
5726 \let\glxtr@inc@linkcount\@glxtr@do@inc@linkcount
5727 \@for\@glxtr@label:=#2\do
5728 {%
5729 \glsssetcategoryattribute{\@glxtr@label}{linkcount}{true}%
5730 \ifstrempy{#1}{}%
5731 {%
5732 \ifcsundef{c@#1}%
5733 {\@nocounterr{#1}}%
5734 {\glsssetcategoryattribute{\@glxtr@label}{linkcountmaster}{#1}}%
5735 }%
5736 }%
5737 }
5738 \@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.

```
5739 \@ifpackageloaded{glossaries-accsupp}
5740 {
```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```
5741 \newcommand*{\glsaccessname}[1]{%
5742   \glsnameaccessdisplay
5743   {%
5744     \glsentryname{#1}%
5745   }%
5746   {#1}%
5747 }
```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```
5748 \newcommand*{\Glsaccessname}[1]{%
5749   \glsnameaccessdisplay
5750   {%
5751     \Glsentryname{#1}%
5752   }%
5753   {#1}%
5754 }
```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```
5755 \newcommand*{\GLSaccessname}[1]{%
5756   \glsnameaccessdisplay
5757   {%
5758     \mfirstucMakeUppercase{\glsentryname{#1}}%
5759   }%
5760   {#1}%
5761 }
```

`\glsaccesstext` Display the text value (no link and no check for existence).

```
5762 \newcommand*{\glsaccesstext}[1]{%
5763   \glstextaccessdisplay
5764   {%
5765     \glsentrytext{#1}%
5766   }%
5767   {#1}%
5768 }
```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```
5769 \newcommand*{\Glsaccesstext}[1]{%
5770   \glstextaccessdisplay
5771   {%
5772     \Glsentrytext{#1}%
5773   }%
5774   {#1}%
5775 }
```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```

5776 \newcommand*{\GLSaccessstext}[1]{%
5777   \glstextaccessdisplay
5778   {%
5779     \mfirstucMakeUppercase{\glstentrytext{#1}}}%
5780   }%
5781   {#1}%
5782 }

```

glsaccessplural Display the plural value (no link and no check for existence).

```

5783 \newcommand*{\glsaccessplural}[1]{%
5784   \glspluralaccessdisplay
5785   {%
5786     \glstentryplural{#1}%
5787   }%
5788   {#1}%
5789 }

```

Glsaccessplural Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

5790 \newcommand*{\Glsaccessplural}[1]{%
5791   \glspluralaccessdisplay
5792   {%
5793     \Glstentryplural{#1}%
5794   }%
5795   {#1}%
5796 }

```

GLSaccessplural Display the plural value (no link and no check for existence) converted to upper case.

```

5797 \newcommand*{\GLSaccessplural}[1]{%
5798   \glspluralaccessdisplay
5799   {%
5800     \mfirstucMakeUppercase{\glstentryplural{#1}}}%
5801   }%
5802   {#1}%
5803 }

```

\glsaccessfirst Display the first value (no link and no check for existence).

```

5804 \newcommand*{\glsaccessfirst}[1]{%
5805   \glsfirstaccessdisplay
5806   {%
5807     \glstentryfirst{#1}%
5808   }%
5809   {#1}%
5810 }

```

\Glsaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

5811 \newcommand*{\Glsaccessfirst}[1]{%

```

```

5812   \glsfirstaccessdisplay
5813   {%
5814       \Glsentryfirst{#1}%
5815   }%
5816   {#1}%
5817 }

```

`\GLSaccessfirst` Display the first value (no link and no check for existence) converted to upper case.

```

5818   \newcommand*{\GLSaccessfirst}[1]{%
5819       \glsfirstaccessdisplay
5820       {%
5821           \mfirstucMakeUppercase{\glentryfirst{#1}}%
5822       }%
5823       {#1}%
5824   }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```

5825   \newcommand*{\glsaccessfirstplural}[1]{%
5826       \glsfirstpluralaccessdisplay
5827       {%
5828           \glentryfirstplural{#1}%
5829       }%
5830       {#1}%
5831   }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

5832   \newcommand*{\Glsaccessfirstplural}[1]{%
5833       \glsfirstpluralaccessdisplay
5834       {%
5835           \Glsentryfirstplural{#1}%
5836       }%
5837       {#1}%
5838   }

```

`cessfirstplural` Display the firstplural value (no link and no check for existence) converted to upper case.

```

5839   \newcommand*{\GLSaccessfirstplural}[1]{%
5840       \glsfirstpluralaccessdisplay
5841       {%
5842           \mfirstucMakeUppercase{\glentryfirstplural{#1}}%
5843       }%
5844       {#1}%
5845   }

```

`glsaccesssymbol` Display the symbol value (no link and no check for existence).

```

5846   \newcommand*{\glsaccesssymbol}[1]{%
5847       \glssymbolaccessdisplay
5848       {%

```

```

5849     \glentrysymbol{#1}%
5850   }%
5851   {#1}%
5852 }

```

Glsaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

5853 \newcommand*\Glsaccesssymbol[1]{%
5854   \glssymbolaccessdisplay
5855   {%
5856     \glentrysymbol{#1}%
5857   }%
5858   {#1}%
5859 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

5860 \newcommand*\GLSaccesssymbol[1]{%
5861   \glssymbolaccessdisplay
5862   {%
5863     \mfirstucMakeUppercase{\glentrysymbol{#1}}%
5864   }%
5865   {#1}%
5866 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```

5867 \newcommand*\glssymbolplural[1]{%
5868   \glssymbolpluralaccessdisplay
5869   {%
5870     \glentrysymbolplural{#1}%
5871   }%
5872   {#1}%
5873 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

5874 \newcommand*\Glsaccesssymbolplural[1]{%
5875   \glssymbolpluralaccessdisplay
5876   {%
5877     \glentrysymbolplural{#1}%
5878   }%
5879   {#1}%
5880 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence) converted to upper case.

```

5881 \newcommand*\GLSaccesssymbolplural[1]{%
5882   \glssymbolpluralaccessdisplay
5883   {%
5884     \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%

```

```

5885     }%
5886     {#1}%
5887 }

```

`\glsaccessdesc` Display the desc value (no link and no check for existence).

```

5888 \newcommand*{\glsaccessdesc}[1]{%
5889   \glsdescriptionaccessdisplay
5890   {%
5891     \glsentrydesc{#1}%
5892   }%
5893   {#1}%
5894 }

```

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

5895 \newcommand*{\Glsaccessdesc}[1]{%
5896   \glsdescriptionaccessdisplay
5897   {%
5898     \Glsentrydesc{#1}%
5899   }%
5900   {#1}%
5901 }

```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```

5902 \newcommand*{\GLSaccessdesc}[1]{%
5903   \glsdescriptionaccessdisplay
5904   {%
5905     \mfirstucMakeUppercase{\glsentrydesc{#1}}%
5906   }%
5907   {#1}%
5908 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence).

```

5909 \newcommand*{\glsaccessdescplural}[1]{%
5910   \glsdescriptionpluralaccessdisplay
5911   {%
5912     \glsentrydescplural{#1}%
5913   }%
5914   {#1}%
5915 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

5916 \newcommand*{\Glsaccessdescplural}[1]{%
5917   \glsdescriptionpluralaccessdisplay
5918   {%
5919     \Glsentrydescplural{#1}%
5920   }%

```



```

5921     {#1}%
5922 }

```

`\glsaccessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

5923 \newcommand*{\Glsaccessdescplural}[1]{%
5924   \glsdescriptionpluralaccessdisplay
5925   {%
5926     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
5927   }%
5928   {#1}%
5929 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

5930 \newcommand*{\glsaccessshort}[1]{%
5931   \glsshortaccessdisplay
5932   {%
5933     \glsentryshort{#1}%
5934   }%
5935   {#1}%
5936 }

```

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

5937 \newcommand*{\Glsaccessshort}[1]{%
5938   \glsshortaccessdisplay
5939   {%
5940     \Glsentryshort{#1}%
5941   }%
5942   {#1}%
5943 }

```

`\GLSaccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

5944 \newcommand*{\GLSaccessshort}[1]{%
5945   \glsshortaccessdisplay
5946   {%
5947     \mfirstucMakeUppercase{\glsentryshort{#1}}%
5948   }%
5949   {#1}%
5950 }

```

`\glsaccessshortpl` Display the short plural form (no link and no check for existence).

```

5951 \newcommand*{\glsaccessshortpl}[1]{%
5952   \glsshortpluralaccessdisplay
5953   {%
5954     \glsentryshortpl{#1}%
5955   }%
5956   {#1}%
5957 }

```

`\lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

5958 \newcommand*\lsaccessshortpl[1]{%
5959   \glsshortpluralaccessdisplay
5960   {%
5961     \Glsentryshortpl{#1}%
5962   }%
5963   {#1}%
5964 }

```

`\LSaccessshortpl` Display the shortplural value (no link and no check for existence) converted to upper case.

```

5965 \newcommand*\LSaccessshortpl[1]{%
5966   \glsshortpluralaccessdisplay
5967   {%
5968     \mfirstucMakeUppercase{\Glsentryshortpl{#1}}%
5969   }%
5970   {#1}%
5971 }

```

`\glsaccesslong` Display the long form (no link and no check for existence).

```

5972 \newcommand*\glsaccesslong[1]{%
5973   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5974 }

```

`\Glsaccesslong` Display the long form (no link and no check for existence).

```

5975
5976 \newcommand*\Glsaccesslong[1]{%
5977   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
5978 }

```

`\GLSaccesslong` Display the long value (no link and no check for existence) converted to upper case.

```

5979 \newcommand*\GLSaccesslong[1]{%
5980   \glslongaccessdisplay
5981   {%
5982     \mfirstucMakeUppercase{\Glsentrylong{#1}}%
5983   }%
5984   {#1}%
5985 }

```

`\glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

5986 \newcommand*\glsaccesslongpl[1]{%
5987   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5988 }

```

`\Glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

5989
5990 \newcommand*\Glsaccesslongpl[1]{%
5991   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
5992 }

```

GLSaccesslongpl Display the longplural value (no link and no check for existence) converted to upper case.

```

5993 \newcommand*{\GLSaccesslongpl}[1]{%
5994   \glslongpluralaccessdisplay
5995   {%
5996     \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
5997   }%
5998   {#1}%
5999 }

```

Keys for accessibility support.

```

6000 \define@key{glsxtrabbrv}{access}{%
6001   \def\@gls@nameaccess{#1}%
6002 }
6003 \define@key{glsxtrabbrv}{textaccess}{%
6004   \def\@gls@textaccess{#1}%
6005 }
6006 \define@key{glsxtrabbrv}{pluralaccess}{%
6007   \def\@gls@pluralaccess{#1}%
6008 }
6009 \define@key{glsxtrabbrv}{firstaccess}{%
6010   \def\@gls@firstaccess{#1}%
6011 }
6012 \define@key{glsxtrabbrv}{firstpluralaccess}{%
6013   \def\@gls@firstpluralaccess{#1}%
6014 }
6015 \define@key{glsxtrabbrv}{shortaccess}{%
6016   \def\@gls@shortaccess{#1}%
6017 }
6018 \define@key{glsxtrabbrv}{shortpluralaccess}{%
6019   \def\@gls@shortaccesspl{#1}%
6020 }
6021 \define@key{glsxtrabbrv}{longaccess}{%
6022   \def\@gls@longaccess{#1}%
6023 }
6024 \define@key{glsxtrabbrv}{shortlonglaccess}{%
6025   \def\@gls@longaccesspl{#1}%
6026 }

```

@initaccesskeys

```

6027 \newcommand*{\@gls@initaccesskeys}{%
6028   \def\@gls@nameaccess{}%
6029   \def\@gls@textaccess{}%
6030   \def\@gls@pluralaccess{}%
6031   \def\@gls@firstaccess{}%
6032   \def\@gls@firstpluralaccess{}%

```

```

6033 \def\@gls@shortaccess{%
6034 \def\@gls@shortaccesspl{%
6035 \def\@gls@longaccess{%
6036 \def\@gls@longaccesspl{%
6037 }

```

ssattribute@set

```
\gls@ifaccessattribute@set{<attribute>}{<true>}{<false>}
```

```

6038 \newcommand*{\@gls@ifaccessattribute@set}[3]{%
6039 \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
6040 {#2}%
6041 {%
6042 \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
6043 {#3}%
6044 {%
6045 \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
6046 {#2}%
6047 {#3}%
6048 }%
6049 }%
6050 }

```

As from glossaries v4.45, the replacement text support has been corrected so that the accessibility support for abbreviations use the “E” (expanded value) element. This should actually contain the long form since it’s supposed to explain the abbreviation. This is a bit redundant on first use for styles like long-short.

aultshortaccess

```
\glsdefaultshortaccess{<long>}{<short>}
```

This command was only introduced to glossaries-accsupp 1.42 so it may not be defined.

```
6051 \def\glsdefaultshortaccess#1#2{#1 (#2)}
```

signactualsetup

```

6052 \newcommand{\glsxtrassignactualsetup}{%
6053 \let\@empty
6054 \let\emph\@firstofone
6055 \let\textbf\@firstofone
6056 \let\textmd\@firstofone
6057 \let\textit\@firstofone
6058 \let\textsl\@firstofone
6059 \let\textsc\@firstofone

```

```

6060 \let\textrm\@firstofone
6061 \let\textsf\@firstofone
6062 \let\texttt\@firstofone
6063 }

```

s@assign@actual

```

6064 \ifdef\pdfstringdef
6065 {
6066   \newcommand{\@gls@assign@actual}{%
6067     \begingroup
6068     \glstrassignactualsetup
6069     \pdfstringdef\@gls@actualshort{\glstrorgshort}%
6070     \pdfstringdef\@gls@actuallong{\glstrorglong}%
6071     \pdfstringdef\@gls@actualshortpl{\@gls@shortpl}%
6072     \pdfstringdef\@gls@actuallongpl{\@gls@longpl}%
6073     \protected@edef\@gls@tmp{\endgroup
6074       \def\noexpand\@gls@actualshort{\expandonce\@gls@actualshort}%
6075       \def\noexpand\@gls@actuallong{\expandonce\@gls@actuallong}%
6076       \def\noexpand\@gls@actualshortpl{\expandonce\@gls@actualshortpl}%
6077       \def\noexpand\@gls@actuallongpl{\expandonce\@gls@actuallongpl}%
6078     }%
6079     \@gls@tmp
6080   }
6081 }
6082 {
6083   \newcommand{\@gls@assign@actual}{%
6084     \begingroup
6085     \glstrassignactualsetup
6086     \protected@edef\@gls@tmp{\endgroup
6087       \def\noexpand\@gls@actualshort{\glstrorgshort}%
6088       \def\noexpand\@gls@actuallong{\glstrorglong}%
6089       \def\noexpand\@gls@actualshortpl{\@gls@shortpl}%
6090       \def\noexpand\@gls@actuallongpl{\@gls@longpl}%
6091     }%
6092     \@gls@tmp
6093   }
6094 }

```

lt@short@access Renamed \@gls@setup@default@access and removed argument since it can be obtained from \glstrorgshort.

@default@access Assign the default value of the shortaccess key. The argument is the short value passed to \newabbreviation. The shortaccess value should explain the abbreviation.

```

6095 \newcommand{\@gls@setup@default@access}{%
6096   \@gls@assign@actual
6097   \ifdefempty\@gls@shortaccess
6098     {%

```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```

6099 \@gls@ifaccessattribute@set{insertdots}%
6100 {%
6101 \expandafter\@glstr@insertdots\expandafter\@gls@actualshort\expandafter
6102 {\@gls@actualshort}%
6103 }%
6104 {%
6105 \ifdefempty\@gls@longaccess
6106 {%
6107 \edef\@gls@shortaccess{\glsdefaultshortaccess
6108 {\expandonce\@gls@actuallong}{\expandonce\@gls@actualshort}}%
6109 }%
6110 {%
6111 \edef\@gls@shortaccess{\glsdefaultshortaccess
6112 {\expandonce\@gls@longaccess}{\expandonce\@gls@actualshort}}%
6113 }%
6114 \eappto\ExtraCustomAbbreviationFields{shortaccess={\@gls@shortaccess},}%

```

If shortaccessplural hasn't been set, assign plural form.

```

6115 \ifdefempty\@gls@shortaccesspl
6116 {%
6117 \@gls@ifaccessattribute@set{aposplural}%
6118 {%
6119 \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
6120 \@gls@actualshort'\glstrabbrvpluralsuffix}%
6121 }%
6122 {%
6123 \@gls@ifaccessattribute@set{noshortplural}%
6124 {%
6125 \let\@gls@shortaccesspl\@gls@shortaccess
6126 }%
6127 {%
6128 \let\@gls@shortaccesspl\@gls@actualshortpl
6129 }%
6130 }%
6131 \ifdefempty\@gls@longaccesspl
6132 {%
6133 \edef\@gls@shortaccesspl{\glsdefaultshortaccess
6134 {\expandonce\@gls@actuallongpl}{\expandonce\@gls@actualshortpl}}%
6135 }%
6136 {%
6137 \edef\@gls@shortaccesspl{\glsdefaultshortaccess
6138 {\expandonce\@gls@longaccesspl}{\expandonce\@gls@actualshort}}%
6139 }%
6140 \eappto\ExtraCustomAbbreviationFields{shortpluralaccess={\@gls@shortaccesspl},}%
6141 }%
6142 {%}%
6143 }%
6144 {%
6145 \ifdefempty\@gls@shortaccesspl
6146 {\let\@gls@shortaccesspl\@gls@shortaccess}%

```

```

6147     {}%
6148   }%

```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```

6149   \ifdefempty\@gls@nameaccess
6150   {%
6151     \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
6152     {%
6153       \eappto\ExtraCustomAbbreviationFields{access={\@gls@shortaccess},}%
6154     }%
6155   }%
6156 }%
6157 {}%

```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```

6158   \ifdefempty\@gls@textaccess
6159   {%
6160     \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
6161     {%
6162       \eappto\ExtraCustomAbbreviationFields{textaccess={\@gls@shortaccess},}%
6163     }%
6164   }%
6165 }%
6166 {}%
6167   \ifdefempty\@gls@pluralaccess
6168   {%
6169     \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
6170     {%
6171       \eappto\ExtraCustomAbbreviationFields{%
6172         pluralaccess={\@gls@shortaccesspl},%
6173       }%
6174     }%
6175   }%
6176 }%
6177 {}%

```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```

6178   \ifdefempty\@gls@firstaccess
6179   {%
6180     \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
6181     {%
6182       \eappto\ExtraCustomAbbreviationFields{firstaccess={\@gls@shortaccess},}%
6183     }%
6184   }%
6185 }%
6186 {}%
6187   \ifdefempty\@gls@firstpluralaccess
6188   {%
6189     \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
6190     {%
6191       \eappto\ExtraCustomAbbreviationFields{%

```

```

6192         firstpluralaccess={\@gls@shortaccesspl},%
6193     }%
6194 }%
6195 {}%
6196 }%
6197 {}%
6198 }

```

Provide hooks for `\setabbreviationstyle` that automatically set the attributes appropriate for the style. If the name is just the short form and the description contains the long form, then it may not be necessary to set `nameshortaccess` but it would depend on the glossary style.

Need to provide `\glsxtr<category>\<field>accsupp` if not already defined.

`provideaccsuppcmd`

```

6199 \newcommand*\glsxtrprovideaccsuppcmd[2]{%
6200   \ifcsundef{glsxtr#1#2accsupp}%
6201     {\csdef{glsxtr#1#2accsupp}{\glsshortaccsupp}}%
6202     {}%
6203 }

```

`abbrSetNoLongAttrs` For styles where the name, first and text are just the abbreviation.

```

6204 \newcommand*\glsxtrAccSuppAbbrSetNoLongAttrs[1]{%
6205   \glssetcategoryattribute{#1}{nameshortaccess}{true}%
6206   \glssetcategoryattribute{#1}{firstshortaccess}{true}%
6207   \glssetcategoryattribute{#1}{textshortaccess}{true}%
6208   \glsxtrprovideaccsuppcmd{#1}{name}%
6209   \glsxtrprovideaccsuppcmd{#1}{first}%
6210   \glsxtrprovideaccsuppcmd{#1}{firstpl}%
6211   \glsxtrprovideaccsuppcmd{#1}{text}%
6212   \glsxtrprovideaccsuppcmd{#1}{plural}%
6213 }

```

`abbrSetFirstLongAttrs` For styles where the name and text are just the abbreviation. The first form may just be long or may be short and long.

```

6214 \newcommand*\glsxtrAccSuppAbbrSetFirstLongAttrs[1]{%
6215   \glssetcategoryattribute{#1}{nameshortaccess}{true}%
6216   \glssetcategoryattribute{#1}{textshortaccess}{true}%
6217   \glsxtrprovideaccsuppcmd{#1}{name}%
6218   \glsxtrprovideaccsuppcmd{#1}{text}%
6219   \glsxtrprovideaccsuppcmd{#1}{plural}%
6220 }

```

`abbrSetTextShortAttrs` For styles where only the text is just the abbreviation. The name and first form may just be long or may be short and long. The name may also be short but followed by the long form in the description.

```

6221 \newcommand*\glsxtrAccSuppAbbrSetTextShortAttrs[1]{%
6222   \glssetcategoryattribute{#1}{textshortaccess}{true}%
6223   \glsxtrprovideaccsuppcmd{#1}{text}%

```



```

6224 \glxtrprovideaccsuppcmd{#1}{plural}%
6225 }

```

etNameShortAttrs For styles where only the name is just the abbreviation. The first and subsequent form may just be long or may be short and long.

```

6226 \newcommand*{\glxtrAccSuppAbbrSetNameShortAttrs}[1]{%
6227 \glsetcategoryattribute{#1}{nameshortaccess}{true}%
6228 \glxtrprovideaccsuppcmd{#1}{name}%
6229 }

```

etNameLongAttrs For styles where the first and text are just the abbreviation. The name may just be long or may be short and long or the name may be short.

```

6230 \newcommand*{\glxtrAccSuppAbbrSetNameLongAttrs}[1]{%
6231 \glsetcategoryattribute{#1}{firstshortaccess}{true}%
6232 \glsetcategoryattribute{#1}{textshortaccess}{true}%
6233 \glxtrprovideaccsuppcmd{#1}{first}%
6234 \glxtrprovideaccsuppcmd{#1}{firstpl}%
6235 \glxtrprovideaccsuppcmd{#1}{text}%
6236 \glxtrprovideaccsuppcmd{#1}{plural}%
6237 }

```

End of if accsupp part

```

6238 }
6239 {

```

No accessibility support. Just define these commands to do \glentry<xxx>

\glsaccessname Display the name value (no link and no check for existence).

```

6240 \newcommand*{\glsaccessname}[1]{\glentryname{#1}}

```

\Glsaccessname Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

6241 \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}

```

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.

```

6242 \newcommand*{\GLSaccessname}[1]{%
6243 \protect\mfirstucMakeUppercase{\glentryname{#1}}}

```

\glsaccesstext Display the text value (no link and no check for existence).

```

6244 \newcommand*{\glsaccesstext}[1]{\glentrytext{#1}}

```

\Glsaccesstext Display the text value (no link and no check for existence) with the first letter converted to upper case.

```

6245 \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}

```

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.

```

6246 \newcommand*{\GLSaccesstext}[1]{%
6247 \protect\mfirstucMakeUppercase{\glentrytext{#1}}}

```

`glsaccessplural` Display the plural value (no link and no check for existence).
6248 `\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.
6249 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.
6250 `\newcommand*{\GLSaccessplural}[1]{%`
6251 `\protect\mfirstucMakeUppercase{\glsentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).
6252 `\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.
6253 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.
6254 `\newcommand*{\GLSaccessfirst}[1]{%`
6255 `\protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}`

`cessfirstplural` Display the firstplural value (no link and no check for existence).
6256 `\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}`

`Glscessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.
6257 `\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}`

`GLScessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.
6258 `\newcommand*{\GLSaccessfirstplural}[1]{%`
6259 `\protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}`

`glsaccesssymbol` Display the symbol value (no link and no check for existence).
6260 `\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.
6261 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.
6262 `\newcommand*{\GLSaccesssymbol}[1]{%`
6263 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`esssymbolplural` Display the symbolplural value (no link and no check for existence).
6264 `\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.

```
6265 \newcommand*{\GLsaccesssymbolplural}[1]{\GLsentrysymbolplural{#1}}
```

esssymbolplural Display the symbolplural value (no link and no check for existence). converted to upper case.

```
6266 \newcommand*{\GLSaccesssymbolplural}[1]{%
6267 \protect\mfirstucMakeUppercase{\glentrysymbolplural{#1}}}
```

\glsaccessdesc Display the desc value (no link and no check for existence).

```
6268 \newcommand*{\glsaccessdesc}[1]{\glentrydesc{#1}}
```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```
6269 \newcommand*{\Glsaccessdesc}[1]{\GLsentrydesc{#1}}
```

\GLSaccessdesc Display the desc value (no link and no check for existence). converted to upper case.

```
6270 \newcommand*{\GLSaccessdesc}[1]{%
6271 \protect\mfirstucMakeUppercase{\glentrydesc{#1}}}
```

ccessdescplural Display the descplural value (no link and no check for existence).

```
6272 \newcommand*{\glsaccessdescplural}[1]{\glentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```
6273 \newcommand*{\Glsaccessdescplural}[1]{\GLsentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence). converted to upper case.

```
6274 \newcommand*{\GLSaccessdescplural}[1]{%
6275 \protect\mfirstucMakeUppercase{\glentrydescplural{#1}}}
```

\glsaccessshort Display the short form (no link and no check for existence).

```
6276 \newcommand*{\glsaccessshort}[1]{\glentryshort{#1}}
```

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```
6277 \newcommand*{\Glsaccessshort}[1]{\GLsentryshort{#1}}
```

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.

```
6278 \newcommand*{\GLSaccessshort}[1]{%
6279 \protect\mfirstucMakeUppercase{\glentryshort{#1}}}
```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```
6280 \newcommand*{\glsaccessshortpl}[1]{\glentryshortpl{#1}}
```

`\lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```
6281 \newcommand*{\lsaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

`\LSaccessshortpl` Display the shortplural value (no link and no check for existence). converted to upper case.

```
6282 \newcommand*{\LSaccessshortpl}[1]{%
6283 \protect\mfirstucMakeUppercase{\Glsentryshortpl{#1}}}
```

`\glsaccesslong` Display the long form (no link and no check for existence).

```
6284 \newcommand*{\glsaccesslong}[1]{\Glsentrylong{#1}}
```

`\Glsaccesslong` Display the long form (no link and no check for existence).

```
6285 \newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}
```

`\GLSaccesslong` Display the long value (no link and no check for existence). converted to upper case.

```
6286 \newcommand*{\GLSaccesslong}[1]{%
6287 \protect\mfirstucMakeUppercase{\Glsentrylong{#1}}}
```

`\glsaccesslongpl` Display the long plural form (no link and no check for existence).

```
6288 \newcommand*{\glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

`\Glsaccesslongpl` Display the long plural form (no link and no check for existence).

```
6289 \newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

`\GLSaccesslongpl` Display the longplural value (no link and no check for existence). converted to upper case.

```
6290 \newcommand*{\GLSaccesslongpl}[1]{%
6291 \protect\mfirstucMakeUppercase{\Glsentrylongpl{#1}}}
```

`@initaccesskeys` This does nothing if there's no accessibility support.

```
6292 \newcommand*{@gls@initaccesskeys}{}%
```

`@default@access` This does nothing if there's no accessibility support.

```
6293 \newcommand{@gls@setup@default@access}{}%
```

`\rSetNoLongAttrs` This does nothing if there's no accessibility support.

```
6294 \newcommand*{\glsxtrAccSuppAbbrSetNoLongAttrs}[1]{}%
```

`\tFirstLongAttrs` This does nothing if there's no accessibility support.

```
6295 \newcommand*{\glsxtrAccSuppAbbrSetFirstLongAttrs}[1]{}%
```

`\tTextShortAttrs` This does nothing if there's no accessibility support.

```
6296 \newcommand*{\glsxtrAccSuppAbbrSetTextShortAttrs}[1]{}%
```

`\tNameShortAttrs` This does nothing if there's no accessibility support.

```
6297 \newcommand*{\glsxtrAccSuppAbbrSetNameShortAttrs}[1]{}%
```

`\tNameLongAttrs` This does nothing if there's no accessibility support.

```
6298 \newcommand*{\glsxtrAccSuppAbbrSetNameLongAttrs}[1]{}%
```

End of else part

```
6299 }
```

1.6 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```
6300 \glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```
6301 \newcommand{\glsifcategory}[4]{%
6302   \ifglstfield{#1}{category}{#2}{#3}{#4}%
6303 }
```

Categories can have attributes.

categoryattribute

`\glssetcategoryattribute{<category>}{<attribute-label>}{<value>}`

Set (or override if already set) an attribute for the given category.

```
6304 \newcommand*{\glssetcategoryattribute}[3]{%
6305   \csdef{@glstr@categoryattr@@#1@#2}{#3}%
6306 }
```

categoryattribute

`\glsgetcategoryattribute{<category>}{<attribute-label>}`

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
6307 \newcommand*{\glsgetcategoryattribute}[2]{%
6308   \csuse{@glstr@categoryattr@@#1@#2}%
6309 }
```

categoryattribute

`\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}`

Tests if the category has the given attribute set.

```
6310 \newcommand*{\glshascategoryattribute}[4]{%
6311   \ifcsvoid{@glstr@categoryattr@@#1@#2}{#4}{#3}%
6312 }
```

glissetattribute

```
\glsssetattribute{<entry label>}{<attribute-label>}{<value>}
```

Short cut where the category label is obtained from the entry information.

```
6313 \newcommand*{\glsssetattribute}[3]{%
6314   \glsssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
6315 }
```

glssgetattribute

```
\glssgetattribute{<entry label>}{<attribute-label>}
```

Short cut where the category label is obtained from the entry information.

```
6316 \newcommand*{\glssgetattribute}[2]{%
6317   \glssgetcategoryattribute{\glscategory{#1}}{#2}%
6318 }
```

glsshhasattribute

```
\glsshhasattribute{<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.

```
6319 \newcommand*{\glsshhasattribute}[4]{%
6320   \ifglssentryexists{#1}%
6321   {\glsshcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
6322   {#4}%
6323 }
```

glsscategoryattribute

```
\glssifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}
```

True if category has the attribute with the given value.

```
6324 \newcommand{\glssifcategoryattribute}[5]{%
6325   \ifcsundef{@glssxtr@categoryattr@@#1@#2}%
6326   {#5}%
6327   {\ifcsstring{@glssxtr@categoryattr@@#1@#2}{#3}{#4}{#5}}%
6328 }
```

\glssifattribute

```
\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.

```
6329 \newcommand{\glsifattribute}[5]{%
6330   \ifglentryexists{#1}%
6331   {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
6332   {#5}%
6333 }
```

Set attributes for the default general category:

```
6334 \glssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
6335 \glssetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to add the regular attribute.

```
6336 \newcommand*{\glssetregularcategory}[1]{%
6337   \glssetcategoryattribute{#1}{regular}{true}%
6338 }
```

`regularcategory`

```
\glsifregularcategory{<category>}{<true part>}{<false part>}
```

Short cut to determine if a category has the regular attribute explicitly set to true.

```
6339 \newcommand{\glsifregularcategory}[3]{%
6340   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
6341 }
```

`regularcategory`

```
\glsifnotregularcategory{<category>}{<true part>}{<false part>}
```

Short cut to determine if a category has the regular attribute explicitly set to false.

```
6342 \newcommand{\glsifnotregularcategory}[3]{%
6343   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
6344 }
```

`\glsifregular`

```
\glsifregular{<entry label>}{<true part>}{<false part>}
```

Short cut to determine if an entry has a regular attribute set to true.

```

6345 \newcommand{\glsifregular}[3]{%
6346   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
6347 }

```

`glsifnotregular`

```
\glsifnotregular{<entry label>}{<true part>}{<false part>}
```

Short cut to determine if an entry has a regular attribute set to false.

```

6348 \newcommand{\glsifnotregular}[3]{%
6349   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
6350 }

```

`foreachincategory`

```
\glsforeachincategory[<glossary labels>]{<category-label>}
{<glossary-cs>}{<label-cs>}{<body>}
```

Iterates through all entries in all the glossaries (or just those listed in *<glossary labels>*) and does *<body>* if the category matches *<category-label>*. The control sequences *<glossary-cs>* and *<label-cs>* may be used in *<body>* to access the glossary label and entry label for the current iteration.

```

6351 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
6352   \forallglossaries[#1]{#3}%
6353   {%
6354     \forglentries[#3]{#4}%
6355     {%
6356       \glsifcategory{#4}{#2}{#5}{%
6357       }%
6358     }%
6359 }

```

`chwithattribute`

```
\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.

```

6360 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
6361   \forallglossaries[#1]{#4}%

```



```

6362  {%
6363    \forglsentries[#4]{#5}%
6364    {%
6365      \glsifattribute{#5}{#2}{#3}{#6}{}%
6366    }%
6367  }%
6368 }

```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to `index` and add `\glsxtrpostdescription`.

```

6369 \ifdef\newterm
6370 {%

```

`\newterm`

```

6371  \renewcommand*{\newterm}[2][]{%
6372    \newglossaryentry{#2}%
6373    {type={index},category=index,name={#2},%
6374     description={\glsxtrpostdescription\nopostdesc},#1}%
6375  }

```

Indexed terms are regular by default.

```

6376  \glssetcategoryattribute{index}{regular}{true}

```

`trpostdescindex`

```

6377  \newcommand*{\glsxtrpostdescindex}{}

6378 }
6379 {}

```

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`.

```

6380 \ifdef\printsymbols
6381 {%

```

`glsxtrnewsymbol` Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```

6382  \newcommand*{\glsxtrnewsymbol}[3][]{%
6383    \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
6384  }

```

Symbols are regular by default.

```

6385  \glssetcategoryattribute{symbol}{regular}{true}

```

`trpostdescsymbol`

```

6386  \newcommand*{\glsxtrpostdescsymbol}{}

6387 }
6388 {}

```

Similar for the numbers option.

```
6389 \ifdef\printnumbers
6390 {%
```

glsxtrnewnumber

```
6391 \ifdef\printnumbers
6392   \newcommand*\glsxtrnewnumber[3][]{%
6393     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
6394   }
```

Numbers are regular by default.

```
6395   \glssetcategoryattribute{number}{regular}{true}
```

rpostdescnumber

```
6396   \newcommand*\glsxtrpostdescnumber{}
6397 }
6398 {}
```

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.

```
6399 \newcommand*\glsxtrsetcategory[2]{%
6400   \@for\@glsxtr@label:=#1\do
6401   {%
6402     \glsfieldxddef{\@glsxtr@label}{category}{#2}%
6403   }%
6404 }
```

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.

```
6405 \newcommand*\glsxtrsetcategoryforall[2]{%
6406   \forallglossaries[#1]{\@glsxtr@type}{%
6407     \forglentries[\@glsxtr@type]{\@glsxtr@label}%
6408     {%
6409       \glsfieldxddef{\@glsxtr@label}{category}{#2}%
6410     }%
6411   }%
6412 }
```

rfieldtitlecase

`\glsxtrfieldtitlecase{<label>}{<field>}`

Apply title casing to the contents of the given field.

```
6413 \newcommand*\glsxtrfieldtitlecase[2]{%
6414   \expandafter\glsxtrfieldtitlecasesecs\expandafter
6415   {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
6416 }
```

fieldtitlecasecs The command used by \glxtrfieldtitlecase. May be redefined to use a different command, for example, \xcapitalisefmtwords.

```
6417 \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.

```
6418 \ifpackageloaded{glossaries-accsupp}
6419 {
6420   \renewcommand*{\glossentrydesc}[1]{%
6421     \glsoifexistsorwarn{#1}%
6422     {%
6423       \glissetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```
6424   \glshasattribute{#1}{glossdescfont}%
6425   {%
6426     \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
6427     \ifcsdef{\@glxtr@attrval}%
6428     {%
6429       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6430     }%
6431     {%
6432       \GlossariesExtraWarning{Unknown control sequence name
6433         ‘\@glxtr@attrval’ supplied in glossdescfont attribute
6434         for entry ‘#1’. Ignoring}%
6435       \let\@glxtr@glossdescfont\@firstofone
6436     }%
6437   }%
6438   {\let\@glxtr@glossdescfont\@firstofone}%
6439   \glisifattribute{#1}{glossdesc}{firstuc}%
6440   {%
6441     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
6442   }%
6443   {%
6444     \glisifattribute{#1}{glossdesc}{title}%
6445     {%
6446       \@glxtr@do@titlecaps@warn
6447       \glsddescriptionaccessdisplay
6448     {%
6449       \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
6450     }%
6451     {#1}%
6452   }%
6453   {%
6454     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
6455   }%
```

```

6456     }%
6457 }%
6458 }
6459 }
6460 {
6461   \renewcommand*{\glossentrydesc}[1]{%
6462     \glsdoifexistsorwarn{#1}%
6463     {%
6464       \glssetabbrvfmt{\glscategory{#1}}%
6465       \glsattribute{#1}{glossdescfont}%
6466       {%
6467         \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
6468         \ifcsdef{\@glxtr@attrval}%
6469         {%
6470           \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
6471         }%
6472         {%
6473           \GlossariesExtraWarning{Unknown control sequence name
6474             '\@glxtr@attrval' supplied in glossdescfont attribute
6475             for entry '#1'. Ignoring}%
6476           \let\@glxtr@glossdescfont\@firstofone
6477         }%
6478       }%
6479       {\let\@glxtr@glossdescfont\@firstofone}%
6480       \glsifattribute{#1}{glossdesc}{firstuc}%
6481       {%
6482         \@glxtr@glossdescfont{\Glsentrydesc{#1}}%
6483       }%
6484       {%
6485         \glsifattribute{#1}{glossdesc}{title}%
6486         {%
6487           \@glxtr@do@titlecaps@warn
6488           \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
6489         }%
6490         {%
6491           \@glxtr@glossdescfont{\glentrydesc{#1}}%
6492         }%
6493       }%
6494     }%
6495   }
6496 }

```

`\glossentryname` If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

6497 \ifpackageloaded{glossaries-accsupp}
6498 {
6499   \renewcommand*{\glossentryname}[1]{%
6500     \glsdoifexistsorwarn{#1}%
6501     {%

```

6502 \glssetabbrvfmt{\glscategory{#1}}%

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

6503       \glsattribute{#1}{glossnamefont}%
6504       {%
6505       \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6506       \ifcsdef{\@glsxtr@attrval}%
6507       {%
6508       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6509       }%
6510       {%
6511       \GlossariesExtraWarning{Unknown control sequence name
6512       '\@glsxtr@attrval' supplied in glossnamefont attribute
6513       for entry '#1'. Reverting to default \string\glsnamefont}%
6514       \let\@glsxtr@glossnamefont\glsnamefont
6515       }%
6516       }%
6517       {\let\@glsxtr@glossnamefont\glsnamefont}%
6518       \glsifattribute{#1}{glossname}{firstuc}%
6519       {%
6520       \glsnameaccessdisplay
6521       {%
6522       \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6523       }%
6524       {#1}%
6525       }%
6526       {%
6527       \glsifattribute{#1}{glossname}{title}%
6528       {%
6529       \@glsxtr@do@titlecaps@warn
6530       \glsnameaccessdisplay
6531       {%
6532       \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
6533       }%
6534       {#1}%
6535       }%
6536       {%
6537       \glsifattribute{#1}{glossname}{uc}%
6538       {%
6539       \glsnameaccessdisplay
6540       {%

```

Hide the label from the upper-casing command.

```

6541       \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
6542       \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
6543       }%
6544       {#1}%
6545       }%
6546       {%
6547       \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%

```

```

6548         \glsnameaccessdisplay
6549         {%
6550         \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
6551         }%
6552         {#1}%
6553     }%
6554 }%
6555 }%

```

Do post-name hook:

```

6556     \glstrpostnamehook{#1}%
6557 }%
6558 }
6559 }
6560 {
6561     \renewcommand*{\glossentryname}[1]{%
6562     \@glsdoifexistsorwarn{#1}%
6563     {%
6564         \glsetabbrvfmt{\glscategory{#1}}%
6565         \glshasattribute{#1}{glossnamefont}%
6566         {%
6567             \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
6568             \ifcsdef{\@glstr@attrval}%
6569             {%
6570                 \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
6571             }%
6572             {%
6573                 \GlossariesExtraWarning{Unknown control sequence name
6574                 '\@glstr@attrval' supplied in glossnamefont attribute
6575                 for entry '#1'. Reverting to default \string\glsnamefont}%
6576                 \let\@glstr@glossnamefont\glsnamefont
6577             }%
6578         }%
6579         {\let\@glstr@glossnamefont\glsnamefont}%
6580         \glsifattribute{#1}{glossname}{firstuc}%
6581         {%
6582             \@glstr@glossnamefont{\Glsentryname{#1}}%
6583         }%
6584         {%
6585             \glsifattribute{#1}{glossname}{title}%
6586             {%
6587                 \@glstr@do@titlecaps@warn
6588                 \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}%
6589             }%
6590             {%
6591                 \glsifattribute{#1}{glossname}{uc}%
6592                 {%

```

Hide the label from the upper-casing command.

```

6593         \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%

```

```

6594         \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6595     }%
6596     {%

```

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.

```

6597         \letcs{\glo@name}{glo@glstetoklabel{#1}@name}%
6598         \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
6599     }%
6600 }%
6601 }%

```

Do post-name hook.

```

6602     \glsxtrpostnamehook{#1}%
6603 }%
6604 }
6605 }

```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```

6606 \ifpackageloaded{glossaries-accsupp}
6607 {
6608     \renewcommand*{\Glossentryname}[1]{%
6609         \@glsdoifexistsorwarn{#1}%
6610         {%
6611             \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the `glossnamefont` attribute to determine the font applied.

```

6612     \glshasattribute{#1}{glossnamefont}%
6613     {%
6614         \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6615         \ifcsdef{\@glsxtr@attrval}%
6616         {%
6617             \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6618         }%
6619         {%
6620             \GlossariesExtraWarning{Unknown control sequence name
6621                 '\@glsxtr@attrval' supplied in glossnamefont attribute
6622                 for entry '#1'. Reverting to default \string\glsnamefont}%
6623             \let\@glsxtr@glossnamefont\glsnamefont
6624         }%
6625     }%
6626     {\let\@glsxtr@glossnamefont\glsnamefont}%
6627     \glsnameaccessdisplay
6628     {%
6629         \@glsxtr@glossnamefont{\Glsentryname{#1}}%
6630     }%
6631     {#1}%

```

Do post-name hook:

```

6632     \glsxtrpostnamehook{#1}%
6633 }%

```

```

6634 }
6635 }
6636 {
6637   \renewcommand*{\Glossentryname}[1]{%
6638     \@glsdoifexistsorwarn{#1}%
6639     {%
6640       \glssetabbrvfmt{\glscategory{#1}}%
6641       \glshasattribute{#1}{glossnamefont}%
6642       {%
6643         \edef\@glxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
6644         \ifcsdef{\@glxtr@attrval}%
6645         {%
6646           \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
6647         }%
6648         {%
6649           \GlossariesExtraWarning{Unknown control sequence name
6650             '\@glxtr@attrval' supplied in glossnamefont attribute
6651             for entry '#1'. Reverting to default \string\glsnamefont}%
6652           \let\@glxtr@glossnamefont\glsnamefont
6653         }%
6654       }%
6655       {\let\@glxtr@glossnamefont\glsnamefont}%
6656       \@glxtr@glossnamefont{\Glsentryname{#1}}%

```

Do post-name hook:

```

6657   \glxtrpostnamehook{#1}%
6658 }%
6659 }
6660 }

```

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`\glxtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```

6661 \newcommand*{\glxtrpostnamehook}[1]{%
6662   \let\@glsnumberformat\@glxtr@defaultnumberformat
6663   \glxtrdoautoindexname{#1}{indexname}%

```

Allow additional code regardless of category:

```

6664   \glsextrapostnamehook{#1}%

```

Allow categories to hook in here.

```

6665   \csuse{glxtrpostname\glscategory{#1}}%
6666 }

```

`\glsextrapostnamehook`

```

6667 \newcommand*{\glsextrapostnamehook}[1]{%

```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.


```

6668 \newcommand*{\glsdefpostname}[2]{%
6669   \csdef{glsxtrpostname#1}{#2}%
6670 }

```

etaccessdisplay

```

6671 \@ifpackageloaded{glossaries-accsupp}
6672 {
6673   \newcommand*{\glsxtr@setaccessdisplay}[1]{%
6674     \ifcsdef{gls#1accessdisplay}%
6675       {\letcs\glsxtr@accessdisplay{gls#1accessdisplay}}%
6676       {%

```

This is essentially the reverse of `\@gls@fetchfield`, since the field supplied to `\glossentryname` has to be the internal label, but the `\gls{field}accessdisplay` commands use the key name.

```

6677     \edef\@gls@thisval{#1}%
6678     \@for\@gls@map:=\@gls@keymap\do{%
6679       \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
6680       \ifdefequal{\@this@key}{\@gls@thisval}%
6681       {%
6682         \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
6683         \@endfortrue
6684       }%
6685     }%
6686   }%
6687   \ifcsdef{gls\@gls@thisval accessdisplay}%
6688   {\letcs\glsxtr@accessdisplay{gls\@gls@thisval accessdisplay}}%
6689   {\let\glsxtr@accessdisplay\@firstoftwo}%
6690 }%
6691 }
6692 }
6693 {%
6694   \newcommand*{\glsxtr@setaccessdisplay}[1]{%
6695     \let\glsxtr@accessdisplay\@firstoftwo
6696 }

```

sentrynameother Provide a command that works like `\glossentryname` but accesses a different field (which must be supplied using its internal field label).

```

6697 \newrobustcmd*{\glossentrynameother}[2]{%
6698   \@glsdoifexistsorwarn{#1}%
6699   {%

```

Accessibility support:

```

6700   \glsxtr@setaccessdisplay{#2}%

```

Set the abbreviation format:

```

6701   \glssetabbrvfmt{\glscategory{#1}}%
6702   \glshasattribute{#1}{glossnamefont}%
6703   {%
6704     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%

```

```

6705     \ifcsdef{\@glsxtr@attrval}%
6706     {%
6707         \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
6708     }%
6709     {%
6710         \GlossariesExtraWarning{Unknown control sequence name
6711         '\@glsxtr@attrval' supplied in glossnamefont attribute
6712         for entry '#1'. Reverting to default \string\glsnamefont}%
6713         \let\@glsxtr@glossnamefont\glsnamefont
6714     }%
6715 }%
6716 {\let\@glsxtr@glossnamefont\glsnamefont}%
6717 \glsifattribute{#1}{glossname}{firstuc}%
6718 {%
6719     \@glsxtr@accessdisplay
6720     {\@glsxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
6721     {#1}%
6722 }%
6723 {%
6724     \glsifattribute{#1}{glossname}{title}%
6725     {%
6726         \@glsxtr@do@titlecaps@warn
6727         \@glsxtr@accessdisplay
6728         {\@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{#2}}}%
6729         {#1}%
6730     }%
6731     {%
6732         \glsifattribute{#1}{glossname}{uc}%
6733         {%
6734             \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6735             \@glsxtr@accessdisplay
6736             {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
6737             {#1}%
6738         }%
6739         {%
6740             \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
6741             \@glsxtr@accessdisplay
6742             {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
6743             {#1}%
6744         }%
6745     }%
6746 }%

```

Do post-name hook.

```

6747     \glsxtrpostnamehook{#1}%
6748 }%
6749 }

```

`format@override` Determines if the format key should override the indexing attribute value.

```

6750 \newif\if@glsxtr@format@override

```

```
6751 \@glxstr@format@overridefalse
```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```
6752 \@ifpackageloaded{hyperref}
6753 {
```

If `hyperref`'s `hyperindex` option is on, then `hyperref` will automatically add `\hyperpage`, so don't add it.

```
6754   \ifHy@hyperindex
6755     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6756       \@glxstr@format@override true
6757       \appto\theindex{\let\glshypernumber\@firstofone}%
6758     }
6759   \else
6760     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6761       \@glxstr@format@override true
6762       \appto\theindex{\let\glshypernumber\hyperpage}%
6763     }
6764   \fi
6765 }
6766 {
6767   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
6768     \@glxstr@format@override true
6769   }
6770 }
6771 \@onlypreamble\GlsXtrEnableIndexFormatOverride
```

`doautoindexname`

```
6772 \newcommand*{\glxtrdoautoindexname}[2]{%
6773   \glshasattribute{#1}{#2}%
6774   {%
```

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.

```
6775   \@glxstr@autoindex@setname{#1}%
```

If the attribute value is simply “true” don't add an `encap`, otherwise use the value as the `encap`.

```
6776   \protected@edef\@glxstr@attrval{\glsggetattribute{#1}{#2}}%
6777   \if@glxstr@format@override

6778     \ifx\@glsnnumberformat\@glxstr@defaultnumberformat
6779     \else
6780       \let\@glxstr@attrval\@glsnnumberformat
6781     \fi
6782   \fi
6783   \ifdefstring{\@glxstr@attrval}{true}%
6784   {}%
```

```

6785     {\eappto\@glo@name{\@glsxtr@autoindex@encap\@glsxtr@attrval}}}%
6786     \expandafter\glsxtr@autoindex\expandafter{\@glo@name}%
6787 }%
6788 {}%
6789 }

```

glsxtrautoindex

```

6790 \newcommand*{\glsxtrautoindex}{\index}

```

xtrautoindexesc

```

6791 \newcommand{\glsxtrautoindexesc}{%
6792   \@gls@checkmkidxchars\@glo@sort
6793   \@glsxtr@autoindex@doextra@esc\@glo@sort
6794 }

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

6795 \newcommand*{\@glsxtr@autoindex@setname}[1]{%
6796   \protected@edef\@glo@name{\glsxtrautoindexentry{#1}}%
6797   \glsxtrautoindexassignsort{\@glo@sort}{#1}%
6798   \glsxtrautoindexesc
6799   \epreto\@glo@name{\@glo@sort\@glsxtr@autoindex@at}%
6800 }

```

rautoindexentry Command used for the actual part when auto-indexing.

```

6801 \newcommand*{\glsxtrautoindexentry}[1]{\string\glsentryname{#1}}

```

indexassignsort Used to assign the sort value when auto-indexing.

```

6802 \newcommand*{\glsxtrautoindexassignsort}[2]{%
6803   \glsletentryfield{#1}{#2}{sort}%
6804 }

```

dex@doextra@esc

```

6805 \newcommand*{\@glsxtr@autoindex@doextra@esc}[1]{%
    Escape the escape character unless it has already been escaped.
6806   \ifx\@glsxtr@autoindex@esc\@gls@quotechar
6807   \else
6808     \def\@gls@checkedmkidx{}%
6809     \edef\@glsxtr@checkspch{%
6810       \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
6811       \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
6812       \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6813     \@glsxtr@checkspch
6814     \let#1\@gls@checkedmkidx\relax
6815   \fi

```

Escape actual character unless it has already been escaped.

```

6816   \ifx\@glsxtr@autoindex@at\@gls@actualchar
6817   \else

```

```

6818 \def\@gls@checkedmkidx{%
6819 \edef\@@glsxtr@checkspch{%
6820 \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
6821 \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
6822 \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6823 @@glsxtr@checkspch
6824 \let#1\@gls@checkedmkidx\relax
6825 \fi

```

Escape level character unless it has already been escaped.

```

6826 \ifx\@glsxtr@autoindex@level\@gls@levelchar
6827 \else
6828 \def\@gls@checkedmkidx{%
6829 \edef\@@glsxtr@checkspch{%
6830 \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
6831 \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
6832 \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6833 @@glsxtr@checkspch
6834 \let#1\@gls@checkedmkidx\relax
6835 \fi

```

Escape encap character unless it has already been escaped.

```

6836 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
6837 \else
6838 \def\@gls@checkedmkidx{%
6839 \edef\@@glsxtr@checkspch{%
6840 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
6841 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
6842 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
6843 @@glsxtr@checkspch
6844 \let#1\@gls@checkedmkidx\relax
6845 \fi
6846 }

```

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.

```

6847 \newcommand*{\@glsxtr@autoindex@at}{}

```

trSetActualChar Set the actual character.

```

6848 \newcommand*{\GlsXtrSetActualChar}[1]{%
6849 \gdef\@glsxtr@autoindex@at{#1}%
6850 \def\@glsxtr@autoindex@escat##1##2##3\@glsxtr@endescspch{%
6851 \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
6852 }%
6853 }
6854 \@onlypreamble\GlsXtrSetActualChar
6855 \makeatother
6856 \GlsXtrSetActualChar{@}

```

```

6857 \makeatletter

autoindex@encap  Encap character for use with \index.
6858 \newcommand*{\@glxtr@autoindex@encap}{}

XtrSetEncapChar  Set the encap character.
6859 \newcommand*{\GlsXtrSetEncapChar}[1]{%
6860   \gdef\@glxtr@autoindex@encap{#1}%
6861   \def\@glxtr@autoindex@escencap##1##2##3\@glxtr@endescspch{%
6862     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escencap}{##1}{##2}{##3}%
6863   }%
6864 }
6865 \GlsXtrSetEncapChar{}
6866 \@onlypreamble\GlsXtrSetEncapChar

autoindex@level  Level character for use with \index.
6867 \newcommand*{\@glxtr@autoindex@level}{}

XtrSetLevelChar  Set the encap character.
6868 \newcommand*{\GlsXtrSetLevelChar}[1]{%
6869   \gdef\@glxtr@autoindex@level{#1}%
6870   \def\@glxtr@autoindex@esclevel##1##2##3\@glxtr@endescspch{%
6871     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@esclevel}{##1}{##2}{##3}%
6872   }%
6873 }
6874 \GlsXtrSetLevelChar{}
6875 \@onlypreamble\GlsXtrSetLevelChar

r@autoindex@esc  Escape character for use with \index.
6876 \newcommand*{\@glxtr@autoindex@esc}{"}

lsXtrSetEscChar  Set the escape character.
6877 \newcommand*{\GlsXtrSetEscChar}[1]{%
6878   \gdef\@glxtr@autoindex@esc{#1}%
6879   \def\@glxtr@autoindex@escquote##1##2##3\@glxtr@endescspch{%
6880     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escquote}{##1}{##2}{##3}%
6881   }%
6882 }
6883 \GlsXtrSetEscChar{}
6884 \@onlypreamble\GlsXtrSetEscChar

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:
6885 \ifdef\actualchar
6886   {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
6887 {}

Quote character \quotechar:
6888 \ifdef\quotechar
6889   {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
6890 {}

```

Level character \levelchar:

```
6891 \ifdef\levelchar
6892 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
6893 {}
```

Encap character \encapchar:

```
6894 \ifdef\encapchar
6895 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
6896 {}
```

leto@endescspch

```
6897 \def\@glxtr@gobbleto@endescspch#1\@glxtr@endescspch{}
```

oindex@esc@spch

<code>\@glxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}</code>
--

```
6898 \newcommand*{\@glxtr@autoindex@escspch}[5]{%
6899   \@glstmpb=\expandafter{\@glscmdkidx}%
6900   \toks@={#3}%
6901   \ifx\@nnil#3\relax
6902     \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch#5\@glxtr@endescspch}%
6903   \else
6904     \ifx\@nnil#4\relax
6905       \edef\@glscmdkidx{\the\@glstmpb\the\toks@}%
6906       \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch
6907         #4#5\@glxtr@endescspch}%
6908     \else
6909       \edef\@glscmdkidx{\the\@glstmpb\the\toks@
6910         \@glxtr@autoindex@esc#1}%
6911       \def\@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
6912     \fi
6913   \fi
6914   \@glxtr@checkspch
6915 }
```

\Glossentrydesc Redefine to set the abbreviation format and accessibility support.

```
6916 \renewcommand*{\Glossentrydesc}[1]{%
6917   \glsoifexistsorwarn{#1}%
6918   {%
6919     \glsetabbrvfmt{\glscategory{#1}}%
6920     \Glsaccessdesc{#1}%
6921   }%
6922 }
```

lossentrysymbol Redefine to set the format and accessibility support. Allow for the possibility of being used in a section heading for standalone entry definitions.

```

6923 \ifdef\teorpdfstring
6924 {
6925   \renewcommand*{\glossentrysymbol}[1]{%
6926     \teorpdfstring{\@glossentrysymbol{#1}}{\glentrypdfsymbol{#1}}%
6927   }
6928 }
6929 {
6930   \renewcommand*{\glossentrysymbol}[1]{\@glossentrysymbol{#1}}
6931 }

```

entrypdfsymbol May be redefined to a field that expands to a value that's more suitable for PDF bookmarks.

```

6932 \newcommand{\glentrypdfsymbol}[1]{\glentrysymbol{#1}}

```

lossentrysymbol There are no case-changing attributes as it's less usual for symbols.

```

6933 \newrobustcmd*{\@glossentrysymbol}[1]{%
6934   \glsoifexistsorwarn{#1}%
6935   {%
6936     \begingroup
6937       \glsetabbrvfmt{\glscategory{#1}}%
6938       \glshasattribute{#1}{glosssymbolfont}%
6939       {%
6940         \edef\@glxtr@attrval{\glsggetattribute{#1}{glosssymbolfont}}%
6941         \ifcsdef{\@glxtr@attrval}%
6942         {%
6943           \letcs{\@glxtr@glosssymbolfont}{\@glxtr@attrval}%
6944         }%
6945         {%
6946           \GlossariesExtraWarning{Unknown control sequence name
6947             '\@glxtr@attrval' supplied in glosssymbolfont attribute
6948             for entry '#1'. Ignoring}%
6949           \let\@glxtr@glosssymbolfont\@firstofone
6950         }%
6951       }%
6952       {\let\@glxtr@glosssymbolfont\@firstofone}%
6953       \@glxtr@glosssymbolfont{\glaccesssymbol{#1}}%
6954     \endgroup
6955   }%
6956 }

```

lossentrysymbol Redefine to set the abbreviation format and accessibility support.

```

6957 \renewcommand*{\Glossentrysymbol}[1]{%
6958   \glsoifexistsorwarn{#1}%
6959   {%
6960     \glsetabbrvfmt{\glscategory{#1}}%
6961     \Glsaccesssymbol{#1}%
6962   }%
6963 }

```

Allow initials to be marked but only use the formatting for the tag in the glossary.

`eInitialTagging` Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```
6964 \newcommand*{\GlsXtrEnableInitialTagging}{%
6965   \@ifstar\s@glstr@enabletagging\@glstr@enabletagging
6966 }
6967 \@onlypreamble\GlsXtrEnableInitialTagging
```

`r@enabletagging` Starred version undefines command.

```
6968 \newcommand*{\s@glstr@enabletagging}[2]{%
6969   \undef#2%
6970   \@glstr@enabletagging{#1}{#2}%
6971 }
```

`r@enabletagging` Internal command.

```
6972 \newcommand*{\@glstr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
6973   \@for\@glstr@cat:=#1\do
6974   {%
6975     \ifdefempty\@glstr@cat
6976     {}%
6977     {\glsssetcategoryattribute{\@glstr@cat}{tagging}{true}}%
6978   }%
6979   \newrobustcmd*#2[1]{##1}%
6980   \def\@glstr@taggingcs{#2}%
6981   \renewcommand*\@glstr@activate@initialtagging{%
6982     \let#2\@glstr@tag
6983   }%
6984   \ifundef\@glstr@preglossaryhook
6985   {\GlossariesExtraWarning{Initial tagging requires at least
6986     glossaries.sty v4.19 to work correctly}}%
6987   {}%
6988 }
```

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`

```
6989 \ifundef\mfu@checkword@do
6990 {
6991   \newcommand*{\mfu@checkword@do}[1]{%
6992     \ifdefstring{\mfu@checkword@arg}{#1}%
6993     {%
6994       \let\@mfu@domakefirstuc\@firstofone
6995       \listbreak
6996     }%
6997     {}%
6998   }
```

`\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.

```
6999 \ifundef\mfu@checkword
7000 {
7001   \newcommand{\@glstr@do@titlecaps@warn}{%
7002     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
7003       support not available}%
7004     \let\@glstr@do@titlecaps@warn\relax
7005   }
7006 }
7007 {
7008   \renewcommand*\mfu@checkword[1]{%
7009     \def\mfu@checkword@arg{#1}%
7010     \let\@mfu@do@makefirstuc\makefirstuc
7011     \forlistloop\mfu@checkword@do\@mfu@nocaplist
7012   }
7013 }
7014 }
7015 {}% no patch required
```

`@titlecaps@warn` Do warning if title case not supported.

```
7016 \newcommand*\@glstr@do@titlecaps@warn{}
```

`@initialtagging` Used in `\printglossary` but at least v4.19 of `glossaries` required.

```
7017 \newcommand*\@glstr@activate@initialtagging{}
```

`\@glstr@tag` Definition of tagging command when used in glossary.

```
7018 \newrobustcmd*\@glstr@tag[1]{%
7019   \gl@ifattribute{\gl@currententrylabel}{tagging}{true}%
7020   {\glstr@tagfont{#1}}{#1}%
7021 }
```

`\glstr@tagfont` Used in the glossary.

```
7022 \newcommand*\glstr@tagfont[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.

```
7023 \ifdef\@gl@preglossaryhook
7024 {
7025   \renewcommand*\@gl@preglossaryhook{%
7026     \@glstr@activate@initialtagging
```

Since the `glossaries` are automatically scoped, `\@glstr@org@postdescription` shouldn't already be defined, but check anyway just as a precautionary measure.

```

7027 \ifundef\@glxtr@org@postdescription
7028 {%
7029 \let\@glxtr@org@postdescription\glspostdescription
7030 \renewcommand*{\glspostdescription}{%
7031 \ifglentryexists{\glscurrententrylabel}%
7032 {%
7033 \glxtrpostdescription
7034 \@glxtr@org@postdescription
7035 }%
7036 }%
7037 }%
7038 }%
7039 {}%

```

Enable the options used by \@glxtrp:

```

7040 \glossxtrsetpopts
7041 }%
7042 }
7043 {}

```

`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.

```

7044 \newcommand*{\glxtrpostdescription}{%
7045 \csuse{glxtrpostdesc\glscategory{\glscurrententrylabel}}%
7046 }

```

`postdescgeneral`

```

7047 \newcommand*{\glxtrpostdescgeneral}{}

```

`xtrpostdescterm`

```

7048 \newcommand*{\glxtrpostdescterm}{}

```

`postdescacronym`

```

7049 \newcommand*{\glxtrpostdescacronym}{}

```

`descabbreviation`

```

7050 \newcommand*{\glxtrpostdescabbreviation}{}

```

`\glsdefpostdesc` Provide a convenient command for defining the post-description hook for the given category.

```

7051 \newcommand*{\glsdefpostdesc}[2]{%
7052 \csdef{glxtrpostdesc#1}{#2}%
7053 }

```

`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.

```
7054 \renewcommand*{\glspostlinkhook}{%
7055   \ifglstryexists{\glslabel}{\glsxtrpostlinkhook}{}%
7056 }
```

`xtrpostlinkhook` The entry label should already be stored in `\glslabel` by `\@gls@link`.

```
7057 \newcommand*{\glsxtrpostlinkhook}{%
7058   \glsxtrdiscardperiod{\glslabel}%
7059   {\glsxtrpostlinkendsentence}%
7060   {\glsxtrifcustomdiscardperiod
7061     {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}}%
7062   {\glsxtrpostlink}%
7063 }%
7064 }
```

`omdiscardperiod` Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
7065 \newcommand*{\glsxtrifcustomdiscardperiod}[2]{#2}
```

`\glsxtrpostlink`

```
7066 \newcommand*{\glsxtrpostlink}{%
7067   \csuse{glsxtrpostlink\glscategory{\glslabel}}%
7068 }
```

`\glsdefpostlink` Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite `\glsxtrpostlink`.

```
7069 \newcommand*{\glsdefpostlink}[2]{%
  \ifthenelse is used to ensure that the expanded value is tested. (The category label must
  be fully expandable.)
7070   \ifthenelse{\equal{#1}{}}{%
7071     {\PackageError{glossaries-extra}
7072       {Invalid empty category label in \string\glsdefpostlink}{}}%
7073     {\csdef{glsxtrpostlink#1}{#2}}%
7074 }
```

`linkendsentence` Done by `\glsxtrpostlinkhook` if a full stop is discarded.

```
7075 \newcommand*{\glsxtrpostlinkendsentence}{%
7076   \ifcsdef{glsxtrpostlink\glscategory{\glslabel}}
7077   {%
7078     \csuse{glsxtrpostlink\glscategory{\glslabel}}%
  Put the full stop back.
7079   .\spacefactor\sffcode'\. \relax
7080 }%
7081 {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
7082 \spacefactor\sfcode'\. \relax
7083 }%
7084 }
```

DescOnFirstUse 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.

```
7085 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
7086 \glxtrifwasfirstuse{\space\glxtrparen{\glsaccessdesc{\glslabel}}}{}%
7087 }
```

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.

```
7088 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
7089 \glxtrifwasfirstuse
7090 {%
7091 \ifglshassymbol{\glslabel}%
7092 {\space\glxtrparen{\glsaccesssymbol{\glslabel}}}%
7093 }%
7094 }%
7095 {}%
7096 }
```

1DescOnFirstUse Provide a command for appending the symbol (if defined) and description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
7097 \newcommand*{\glxtrpostlinkAddSymbolDescOnFirstUse}{%
7098 \glxtrifwasfirstuse
7099 {%
7100 \space\glxtrparen
7101 {%
7102 \ifglshassymbol{\glslabel}%
7103 {\glsaccesssymbol{\glslabel}, }%
7104 }%
7105 \glsaccessdesc{\glslabel}%
7106 }%
7107 }%
7108 {}%
7109 }
```

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).

```
7110 \newcommand*{\glxtrdiscardperiod}[3]{%
7111 \glxtrifwasfirstuse
7112 {%
7113 \glsifattribute{#1}{retainfirstuseperiod}{true}%
```

```

7114   {#3}%
7115   {%
7116     \glsifattribute{#1}{discardperiod}{true}%
7117     {%
7118       \glsifplural
7119       {%
7120         \glsifattribute{#1}{pluraldiscardperiod}{true}%
7121         {\glsxtrifperiod{#2}{#3}}%
7122         {#3}%
7123       }%
7124     }%
7125     \glsxtrifperiod{#2}{#3}%
7126   }%
7127 }%
7128 {#3}%
7129 }%
7130 }%
7131 {%
7132   \glsifattribute{#1}{discardperiod}{true}%
7133   {%
7134     \glsifplural
7135     {%
7136       \glsifattribute{#1}{pluraldiscardperiod}{true}%
7137       {\glsxtrifperiod{#2}{#3}}%
7138       {#3}%
7139     }%
7140   }%
7141   \glsxtrifperiod{#2}{#3}%
7142 }%
7143 }%
7144 {#3}%
7145 }%
7146 }

```

`\glsxtrifperiod` Make a convenient user command to check if the next character is a full stop (period). Works like `\@ifstar` but uses `\new@ifnextchar` rather than `\@ifnextchar`

```

7147 \newcommand*\glsxtrifperiod}[1]{\new@ifnextchar.{\@firstoftwo{#1}}

```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

`\glsxtr@punclist` List of characters identified as punctuation marks. (Be careful of babel shorthands!) This doesn't allow for punctuation marks made up from multiple characters (such as ' ').

```

7148 \newcommand*\glsxtr@punclist{.,;?!}

```

`\punctuationmark` Add character to punctuation list.

```

7149 \newcommand*\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}

```

`\punctuationmarks` Reset the punctuation list.

```

7150 \newcommand*\glsxtrsetpunctuationmarks}[1]{\def\glsxtr@punclist{#1}}

```

\glxstrifpunc

`\glxstrifnextpunc{<true part>}{<false part>}`

Test if this is followed by a punctuation mark. (Adapted from \new@ifnextchar.)

```
7151 \newcommand*{\glxstrifnextpunc}[2]{%
7152   \def\reserved@a{#1}%
7153   \def\reserved@b{#2}%
7154   \futurelet\@glspunc@token\glxstr@ifnextpunc
7155 }
```

glxstr@ifnextpunc

```
7156 \newcommand*{\glxstr@ifnextpunc}{%
7157   \glxstr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
7158   \reserved@b
7159 }
```

glxstr@ifpunctoken Test if the token given in the first argument is in the punctuation list.

```
7160 \newcommand*{\glxstr@ifpunctoken}[1]{%
7161   \expandafter\@glxstr@ifpunctoken\expandafter#1\glxstr@punclist\@nnil
7162 }
```

glxstr@ifpunctoken

```
7163 \def\@glxstr@ifpunctoken#1#2{%
7164   \let\reserved@d=#2%
7165   \ifx\reserved@d\@nnil
7166     \let\glxstr@next\@glxstr@notfoundinlist
7167   \else
7168     \ifx#1\reserved@d
7169       \let\glxstr@next\@glxstr@foundinlist
7170     \else
7171       \let\glxstr@next\@glxstr@ifpunctoken
7172     \fi
7173   \fi
7174   \glxstr@next#1%
7175 }
```

glxstr@foundinlist

```
7176 \def\@glxstr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
7177 \def\@glxstr@notfoundinlist#1{\@secondoftwo}
```

glxstr@dopostpunc

```
\glxtrdopostpunc{<code>}
```

If this is followed by a punctuation character, do <code> after the character otherwise do <code> before whatever comes next.

```
7178 \newcommand{\glxtrdopostpunc}[1]{%
7179   \glxtrifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
7180 }
```

@glxtr@swaptwo

```
7181 \newcommand{\@glxtr@swaptwo}[2]{#2#1}
```

1.7 Abbreviations

The “acronym” code from glossaries is misnamed as it’s more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there’s a style for the given category, it needs to be applied by `\newabbreviation`.

```
7182 \define@key{glxtrabbrv}{category}{%
7183   \edef\glscategorylabel{#1}%
7184 }
```

Save the short plural form. This may be needed before the entry is defined.

```
7185 \define@key{glxtrabbrv}{shortplural}{%
7186   \def\@gls@shortpl{#1}%
7187 }
```

Similarly for the long plural form.

```
7188 \define@key{glxtrabbrv}{longplural}{%
7189   \def\@gls@longpl{#1}%
7190 }
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```
7191 \newtoks\glsshortpltok
```

\glslongpltok

```
7192 \newtoks\glslongpltok
```

`\glxtr@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.


```

7193 \newcommand*{\@glsxtr@insertdots}[2]{%
7194   \def#1{}%
7195   \@glsxtr@insert@dotes#1#2\@nnil
7196 }

```

xtr@insert@dotes

```

7197 \newcommand*{\@glsxtr@insert@dotes}[2]{%
7198   \ifx\@nnil#2\relax
7199   \let\@glsxtr@insert@dotes@next\@gobble
7200   \else
7201   \ifx\relax#2\relax
7202   \else
7203     \appto#1{#2.}%
7204   \fi
7205   \let\@glsxtr@insert@dotes@next\@glsxtr@insert@dotes
7206   \fi
7207   \@glsxtr@insert@dotes@next#1%
7208 }

```

Similarly provide a way of replacing spaces with \glsxtrwordsep, which first needs to be defined:

\glsxtrwordsep

```

7209 \newcommand*{\glsxtrwordsep}{\space}

```

Each word is marked with

\glsxtrword

```

7210 \newcommand*{\glsxtrword}[1]{#1}

```

tr@markwordseps

```

7211 \newcommand*{\@glsxtr@markwordseps}[2]{%
7212   \def#1{}%
7213   \@glsxtr@mark@wordseps#1#2 \@nnil
7214 }

```

r@mark@wordseps

```

7215 \def\@glsxtr@mark@wordseps#1#2 #3{%
7216   \ifdefempty{#1}%
7217   {\def#1{\protect\glsxtrword{#2}}}%
7218   {\appto#1{\protect\glsxtrwordsep\protect\glsxtrword{#2}}}%
7219   \ifx\@nnil#3\relax
7220   \let\@glsxtr@mark@wordseps@next\relax
7221   \else
7222   \def\@glsxtr@mark@wordseps@next{%
7223     \@glsxtr@mark@wordseps#1#3}%
7224   \fi
7225   \@glsxtr@mark@wordseps@next
7226 }

```

`newabbreviation` Define a new generic abbreviation.

```
7227 \newcommand*{\newabbreviation}[4] [] {%
7228   \glstr@newabbreviation{#1}{#2}{#3}{#4}%
7229 }
```

`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.)

```
7230 \newcommand*{\glstr@newabbreviation}[4] {%
7231   \glstok{#1}%
7232   \glstok{#2}%
7233   \glstok{#3}%
7234   \glstok{#4}%

```

Save the original short and long values (before attribute settings modify them).

```
7235   \def\glstr@orgshort{#3}%
7236   \def\glstr@orglong{#4}%

```

Provide extra settings for hooks (if modified, this command must end with a comma).

```
7237   \def\ExtraCustomAbbreviationFields{%

```

Initialise accessibility settings if required.

```
7238   \@glstr@initaccesskeys

```

Get the category.

```
7239   \def\glscategorylabel{abbreviation}%

```

Ignore the shortplural and longplural keys.

```
7240   \setkeys*{\glstr@abbrv}[shortplural,longplural]{#1}%

```

Set the abbreviation style.

```
7241   \ifcsdef{@glstr@current@glscategorylabel}%
7242     {%

```

Warning should already have been issued.

```
7243     \let\glstr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
7244     \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
7245     \glstr@applyabbrvstyle{\csname @glstr@current@glscategorylabel\endcsname}%
7246     \let\GlsXtrWarnDeprecatedAbbrStyle\glstr@orgwarndep
7247   }%
7248   {%

```

If no style has been associated with this category, fallback on the style for the abbreviation category.

```
7249     \glstr@applyabbrvstyle{\@glstr@current@abbreviation}%
7250   }%

```

Set the default long plural

```
7251   \def\@glstr@longpl{#4\glsppluralsuffix}%
7252   \let\@glstr@default@longpl\@glstr@longpl

```

Has the markwords attribute been set?

```
7253   \glstr@ifcategoryattribute{\glscategorylabel}{markwords}{true}%
7254   {%

```

```

7255 \glsxtr@keywordseps\@gls@long{#4}%
7256 \expandafter\def\expandafter\@gls@longpl\expandafter
7257 {\@gls@long\glspluralsuffix}%
7258 \let\@gls@default@longpl\@gls@longpl

```

Update \glslongtok.

```

7259 \expandafter\glslongtok\expandafter{\@gls@long}%
7260 }%
7261 {}%

```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```

7262 \glsifcategoryattribute{\gls@categorylabel}{markshortwords}{true}%
7263 {%
7264 \glsxtr@keywordseps\@gls@short{#3}%
7265 }%
7266 {%

```

Has the insertdots attribute been set?

```

7267 \glsifcategoryattribute{\gls@categorylabel}{insertdots}{true}%
7268 {%
7269 \glsxtr@insertdots\@gls@short{#3}%
7270 \appto\@gls@short{\@}%
7271 }%
7272 {\def\@gls@short{#3}}%
7273 }%

```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```

7274 \glsifcategoryattribute{\gls@categorylabel}{aposplural}{true}%
7275 {%
7276 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
7277 '\abbrvpluralsuffix}%
7278 }%
7279 {%

```

Has the noshortplural attribute been set?

```

7280 \glsifcategoryattribute{\gls@categorylabel}{noshortplural}{true}%
7281 {%
7282 \let\@gls@shortpl\@gls@short
7283 }%
7284 {%
7285 \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
7286 \abbrvpluralsuffix}%
7287 }%
7288 }%

```

Update \glsshorttok:

```

7289 \expandafter\glsshorttok\expandafter{\@gls@short}%

```

Hook for further customisation if required:

```

7290 \glsxtrnewabbrevpresetkeyhook{#1}{#2}{#3}%

```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary. Ignore the category key (already obtained).

```
7291 \setkeys*{glsxtrabbrv}[category]{#1}%
```

Save in case required.

```
7292 \let\@gls@org@longpl\@gls@longpl
```

```
7293 \let\@gls@org@shortpl\@gls@shortpl
```

Has the plural been explicitly set?

```
7294 \ifx\@gls@default@longpl\@gls@longpl
```

```
7295 \else
```

Has the markwords attribute been set?

```
7296 \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
```

```
7297 {%
```

```
7298 \expandafter\@glsxtr@keywordseps\expandafter\@gls@longpl\expandafter
```

```
7299 {\@gls@longpl}%
```

```
7300 }%
```

```
7301 {}%
```

```
7302 \fi
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
7303 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
```

```
7304 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
7305 \@gls@setup@default@access
```

Do any extra setup provided by hook:

```
7306 \newabbreviationhook
```

Define this entry:

```
7307 \protected@edef\@do@newglossaryentry{%
```

```
7308 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
7309 {%
```

```
7310 type=\glsxtrabbrvtype,%
```

```
7311 category=abbreviation,%
```

```
7312 short={\the\glsshorttok},%
```

```
7313 shortplural={\the\glsshortpltok},%
```

```
7314 long={\the\glslongtok},%
```

```
7315 longplural={\the\glslongpltok},%
```

```
7316 name={\the\glsshorttok},%
```

```
7317 \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
7318 \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
7319 \the\glskeylisttok
```

```
7320 }%
```

```
7321 }%
```

```
7322 \@do@newglossaryentry
```

Obtain the type and add it to the list of abbreviations.

```
7323 \glsxtr@addabbreviationlist{\glsentrytype{\the\glslabeltok}}%
7324 \GlsXtrPostNewAbbreviation
7325 }
```

evpresetkeyhook Hook for extra stuff in \newabbreviation

```
7326 \newcommand*{\glsxtrnewabbrevpresetkeyhook}[3]{}%
```

NewAbbreviation Hook used by abbreviation styles.

```
7327 \newcommand*{\GlsXtrPostNewAbbreviation}{}%
```

bbreviationhook Hook for use with \newabbreviation.

```
7328 \newcommand*{\newabbreviationhook}{}%
```

reviationFields

```
7329 \newcommand*{\CustomAbbreviationFields}{}%
```

\glsxtrparen For the parenthetical styles.

```
7330 \newcommand*{\glsxtrparen}[1]{(#1)}
```

lsxtrfullformat Full format without case change.

```
7331 \newcommand*{\glsxtrfullformat}[2]{%
7332 \glsfirstlongfont{\glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
7333 \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
7334 }
```

lsxtrfullformat Full format with case change.

```
7335 \newcommand*{\Glsxtrfullformat}[2]{%
7336 \glsfirstlongfont{\Glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
7337 \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshort{#1}}}%
7338 }
```

xtrfullplformat Plural full format without case change.

```
7339 \newcommand*{\glsxtrfullplformat}[2]{%
7340 \glsfirstlongfont{\glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
7341 \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}}}%
7342 }
```

xtrfullplformat Plural full format with case change.

```
7343 \newcommand*{\Glsxtrfullplformat}[2]{%
7344 \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
7345 \glsxtrparen{\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}}}%
7346 }
```

\glsxtrfullsep Separator used by full format is a space by default. The argument is the entry's label.

```
7347 \newcommand*{\glsxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with `\glsentryfull` (for example, first use suppresses the long form or uses a footnote).

`\glsentryfullformat` Full format without case change.
 7348 `\newcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}`

`\glsentryfullformat` Full format with case change.
 7349 `\newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}`

`\glsxtrfullplformat` Plural full format without case change.
 7350 `\newcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}`

`\glsentryfullplformat` Plural full format with case change.
 7351 `\newcommand*{\Glsxtrinlinefullplformat}{\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`
 7352 `\renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}`

`\Glsentryfull`
 7353 `\renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}`

`\glsentryfullpl`
 7354 `\renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}`

`\Glsentryfullpl`
 7355 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}`

`\glsfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.
 7356 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.
 7357 `\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use. This is redefined by the abbreviation styles, as appropriate.
 7358 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\glsabbrvdefaultfont`
 7359 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.
 7360 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`longdefaultfont` Default font changing command used for the long form in commands like `\glxtrlong`.

```
7361 \newcommand*{\glslongdefaultfont}[1]{#1}
```

`lsfirstlongfont` Font changing command used for the long form on first use or in the full format.

```
7362 \newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}
```

`longdefaultfont`

```
7363 \newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}
```

`brvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.

```
7364 \newcommand*{\glxtrabbrvpluralsuffix}{\glspluralsuffix}
```

`brvpluralsuffix` Default plural suffix.

```
7365 \newcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}
```

`\glxtrfull` Full form (no case-change).

```
7366 \newrobustcmd*{\glxtrfull}{\@gls@hyp@opt\@ns@glxtrfull}
```

```
7367 \newcommand*{\ns@glxtrfull}[2][\%]
```

```
7368 \new@ifnextchar[{\@glxtr@full{#1}{#2}}]{\%}
```

```
7369 \def\@glxtr@full{#1}{#2}[\%]
```

```
7370 }
```

`\@glxtr@full` Low-level macro:

```
7371 \def\@glxtr@full#1#2[#3]{\%
```

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

```
7372 \@glxtr@record{#1}{#2}{\glslink}%
```

```
7373 \glsdoifexists{#2}%
```

```
7374 {\%
```

```
7375 \glsssetabbrvfmt{\glscategory{#2}}%
```

```
7376 \let\do@glsc@link@checkfirsthyper\@glsc@link@nocheckfirsthyper
```

```
7377 \let\glsc@plural\@secondoftwo
```

```
7378 \let\glsc@scase\@firstofthree
```

```
7379 \let\glsc@insert\@empty
```

```
7380 \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.

```
7381 \glxtrsetupfulldefs
```

```
7382 \@glsc@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
```

```
7383 }%
```

```
7384 \glspostlinkhook
```

```
7385 }
```

trsetupfulldefs

```
7386 \newcommand*{\glxtrsetupfulldefs}{%
7387   \let\glxtrifwasfirstuse\@firstoftwo
7388 }
```

\Glsxtrfull Full form (first letter uppercase).

```
7389 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\@ns@Glsxtrfull}
7390 \newcommand*{\ns@Glsxtrfull}[2][]{%
7391   \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}{%
7392     {\@Glsxtr@full{#1}{#2}[]}%
7393 }
```

\@Glsxtr@full Low-level macro:

```
7394 \def\@Glsxtr@full#1#2[#3]{%
7395   \glsdoifexists{#2}%
7396   {%
7397     \glssetabbrvfmt{\glscategory{#2}}%
7398     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7399     \let\glsifplural\@secondoftwo
7400     \let\glscapscase\@secondofthree
7401     \let\glsinsert\@empty
7402     \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
7403     \glxtrsetupfulldefs
7404     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7405   }%
7406   \glspostlinkhook
7407 }
```

\GLSxtrfull Full form (all uppercase).

```
7408 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\@ns@GLSxtrfull}
7409 \newcommand*{\ns@GLSxtrfull}[2][]{%
7410   \new@ifnextchar[{\@GLSxtr@full{#1}{#2}}{%
7411     {\@GLSxtr@full{#1}{#2}[]}%
7412 }
```

\@GLSxtr@full Low-level macro:

```
7413 \def\@GLSxtr@full#1#2[#3]{%
7414   \glsdoifexists{#2}%
7415   {%
7416     \glssetabbrvfmt{\glscategory{#2}}%
7417     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7418     \let\glsifplural\@secondoftwo
7419     \let\glscapscase\@thirdofthree
7420     \let\glsinsert\@empty
7421     \def\glscustomtext{\mfirstucMakeUppercase{\glxtrinlinefullformat{#2}{#3}}}%
7422     \glxtrsetupfulldefs
7423     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7424   }%
7425   \glspostlinkhook
```


7426 }

`\glsxtrfullpl` Plural full form (no case-change).

```
7427 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\@ns@glsxtrfullpl}
7428 \newcommand*\ns@glsxtrfullpl[2][\%
7429   \new@ifnextchar[\@glsxtr@fullpl{#1}{#2}]]%
7430   {\@glsxtr@fullpl{#1}{#2}[]}%
7431 }
```

`\@glsxtr@fullpl` Low-level macro:

```
7432 \def\@glsxtr@fullpl#1#2[#3]{%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7433   \@glsxtr@record{#1}{#2}{glslink}%
7434   \glsdoifexists{#2}%
7435   {%
7436     \glssetabbrvfmt{\glscategory{#2}}%
7437     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7438     \let\glsifplural\@firstoftwo
7439     \let\glscapscase\@firstofthree
7440     \let\glsinsert\@empty
7441     \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
7442     \glsxtrsetupfulldefs
7443     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7444   }%
7445   \glspostlinkhook
7446 }
```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```
7447 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
7448 \newcommand*\ns@Glsxtrfullpl[2][\%
7449   \new@ifnextchar[\@Glsxtr@fullpl{#1}{#2}]]%
7450   {\@Glsxtr@fullpl{#1}{#2}[]}%
7451 }
```

`\@Glsxtr@fullpl` Low-level macro:

```
7452 \def\@Glsxtr@fullpl#1#2[#3]{%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7453   \@glsxtr@record{#1}{#2}{glslink}%
7454   \glsdoifexists{#2}%
7455   {%
7456     \glssetabbrvfmt{\glscategory{#2}}%
7457     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7458     \let\glsifplural\@firstoftwo
7459     \let\glscapscase\@secondofthree
7460     \let\glsinsert\@empty
```

```

7461 \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
7462 \glstrsetupfulldefs
7463 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7464 }%
7465 \glspostlinkhook
7466 }

```

`\Glsxtrfullpl` Plural full form (all upper case).

```

7467 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\ns@Glsxtrfullpl}
7468 \newcommand*\ns@Glsxtrfullpl[2][]{%
7469 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
7470 {\@Glsxtr@fullpl{#1}{#2}[]}%
7471 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

7472 \def\@Glsxtr@fullpl#1#2[#3]{%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7473 \@glsxtr@record{#1}{#2}{glslink}%
7474 \glsoifexists{#2}%
7475 {%
7476 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7477 \let\glsifplural\@firstoftwo
7478 \let\glscapscase\@thirdofthree
7479 \let\glsinsert\@empty
7480 \def\glscustomtext{%
7481 \mfirstucMakeUppercase{\glxtrinlinefullplformat{#2}{#3}}}%
7482 \glstrsetupfulldefs
7483 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7484 }%
7485 \glspostlinkhook
7486 }

```

The short and long forms work in a similar way to acronyms.

`\glstrshort`

```

7487 \newrobustcmd*{\glstrshort}{\@gls@hyp@opt\ns@glstrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
7488 \newcommand*\ns@glstrshort[2][]{%
7489 \new@ifnextchar[{\@glstrshort{#1}{#2}}{\@glstrshort{#1}{#2}[]}%
7490 }
    Read in the final optional argument:
7491 \def\@glstrshort#1#2[#3]{%
    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7492 \@glsxtr@record{#1}{#2}{glslink}%
7493 \glsoifexists{#2}%
7494 {%

```

Need to make sure \glsabbrvfont is set correctly.

```

7495 \glssetabbrvfmt{\glscategory{#2}}%
7496 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7497 \let\glxtrifwasfirstuse\@secondoftwo
7498 \let\gl@ifplural\@secondoftwo
7499 \let\glscapscase\@firstofthree
7500 \let\glinsert\@empty
7501 \def\glscustomtext{%
7502   \glsabbrvfont{\glsaccessshort{#2}\ifglxtrininsertinside#3\fi}%
7503   \ifglxtrininsertinside\else#3\fi
7504 }%
7505 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7506 }%
7507 \glspostlinkhook
7508 }

```

\Glsxtrshort

```

7509 \newrobustcmd*{\Glsxtrshort}{\@gl@hyp@opt\@ns@Glsxtrshort}

```

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

```

7510 \newcommand*{\ns@Glsxtrshort}[2] [] {%
7511   \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2} []}%
7512 }

```

Read in the final optional argument:

```

7513 \def\@Glsxtrshort#1#2[#3] {%

```

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

```

7514 \@glxtr@record{#1}{#2}{glslink}%
7515 \glsoifexists{#2}%
7516 {%
7517   \glssetabbrvfmt{\glscategory{#2}}%
7518   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7519   \let\glxtrifwasfirstuse\@secondoftwo
7520   \let\gl@ifplural\@secondoftwo
7521   \let\glscapscase\@secondofthree
7522   \let\glinsert\@empty
7523   \def\glscustomtext{%
7524     \glsabbrvfont{\Glsaccessshort{#2}\ifglxtrininsertinside#3\fi}%
7525     \ifglxtrininsertinside\else#3\fi
7526   }%
7527   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7528 }%
7529 \glspostlinkhook
7530 }

```

\GLSxtrshort

```

7531 \newrobustcmd*{\GLSxtrshort}{\@gl@hyp@opt\@ns@GLSxtrshort}

```

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

```
7532 \newcommand*{\ns@GLSxtrshort}[2] [] {%
7533   \new@ifnextchar[{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2} []}%
7534 }
```

Read in the final optional argument:

```
7535 \def\@GLSxtrshort#1#2[#3] {%
```

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

```
7536   \@glxtr@record{#1}{#2}{glslink}%
7537   \glstoifexists{#2}%
7538   {%
7539     \glsssetabbrvfmt{\glscategory{#2}}%
7540     \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7541     \let\glxtrifwasfirstuse\@secondoftwo
7542     \let\glslifplural\@secondoftwo
7543     \let\glscapscase\@thirdofthree
7544     \let\glslinsert\@empty
7545     \def\glscustomtext{%
7546       \mfirstucMakeUppercase
7547       {\glslabbrvfont{\glslaccessshort{#2}\ifglxtrininsertinside#3\fi}%
7548       \ifglxtrininsertinside\else#3\fi
7549     }%
7550   }%
7551   \@glsl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7552   }%
7553   \glspostlinkhook
7554 }
```

\glxtrlong

```
7555 \newrobustcmd*{\glxtrlong}{\@glsl@hyp@opt\ns@glxtrlong}
```

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

```
7556 \newcommand*{\ns@glxtrlong}[2] [] {%
7557   \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
7558 }
```

Read in the final optional argument:

```
7559 \def\@glxtrlong#1#2[#3] {%
```

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

```
7560   \@glxtr@record{#1}{#2}{glslink}%
7561   \glstoifexists{#2}%
7562   {%
7563     \let\do@glsl@link@checkfirsthyper\@glsl@link@nocheckfirsthyper
7564     \let\glxtrifwasfirstuse\@secondoftwo
7565     \let\glslifplural\@secondoftwo
7566     \let\glscapscase\@firstofthree
7567     \let\glslinsert\@empty
```

```

7568 \def\glscustomtext{%
7569 \glslongfont{\glsaccesslong{#2}\ifglxtrinsertinside#3\fi}%
7570 \ifglxtrinsertinside\else#3\fi
7571 }%
7572 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7573 }%
7574 \glspostlinkhook
7575 }

```

\Glsxtrlong

```

7576 \newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\@ns@Glsxtrlong}

Define the un-starred form. Need to determine if there is a final optional argument
7577 \newcommand*{\ns@Glsxtrlong}[2][{}]{%
7578 \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2}[]}%
7579 }

Read in the final optional argument:
7580 \def\@Glsxtrlong#1#2[#3]{%

If the record option has been used, the information needs to be written to the aux file regard-
less of whether the entry exists (unless indexing has been switched off).
7581 \@glsxtr@record{#1}{#2}{glslink}%
7582 \glsdoifexists{#2}%
7583 {%
7584 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7585 \let\glsxtrifwasfirstuse\@secondoftwo
7586 \let\glsifplural\@secondoftwo
7587 \let\glsapscase\@secondofthree
7588 \let\glsinsert\@empty
7589 \def\glscustomtext{%
7590 \glslongfont{\Glsaccesslong{#2}\ifglxtrinsertinside#3\fi}%
7591 \ifglxtrinsertinside\else#3\fi
7592 }%
7593 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7594 }%
7595 \glspostlinkhook
7596 }

```

\GLSxtrlong

```

7597 \newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\@ns@GLSxtrlong}

Define the un-starred form. Need to determine if there is a final optional argument
7598 \newcommand*{\ns@GLSxtrlong}[2][{}]{%
7599 \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2}[]}%
7600 }

Read in the final optional argument:
7601 \def\@GLSxtrlong#1#2[#3]{%

```

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

```

7602 \@glsxtr@record{#1}{#2}{glslink}%
7603 \glsdoifexists{#2}%
7604 {%
7605   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7606   \let\glsxtrifwasfirstuse\@secondoftwo
7607   \let\glsifplural\@secondoftwo
7608   \let\glscapscase\@thirdofthree
7609   \let\glsinsert\@empty
7610   \def\glscustomtext{%
7611     \mfirstucMakeUppercase
7612     {\glslongfont{\glsaccessslong{#2}\ifglsxtrininsertinside#3\fi}%
7613     \ifglsxtrininsertinside\else#3\fi
7614   }%
7615 }%
7616 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7617 }%
7618 \glspostlinkhook
7619 }

```

Plural short forms:

\glsxtrshortpl

```

7620 \newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\@ns@glsxtrshortpl}

```

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

```

7621 \newcommand*{\ns@glsxtrshortpl}[2] [] {%
7622   \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2} []}%
7623 }

```

Read in the final optional argument:

```

7624 \def\@glsxtrshortpl#1#2[#3] {%

```

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

```

7625 \@glsxtr@record{#1}{#2}{glslink}%
7626 \glsdoifexists{#2}%
7627 {%
7628   \glssetabbrvfmt{\glscategory{#2}}%
7629   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7630   \let\glsxtrifwasfirstuse\@secondoftwo
7631   \let\glsifplural\@firstoftwo
7632   \let\glscapscase\@firstofthree
7633   \let\glsinsert\@empty
7634   \def\glscustomtext{%
7635     \glsabbrvfont{\glsaccesssshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7636     \ifglsxtrininsertinside\else#3\fi
7637   }%
7638   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%

```

```

7639 }%
7640 \glspostlinkhook
7641 }

```

\Glsxtrshortpl

```

7642 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\ns@Glsxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7643 \newcommand*{\ns@Glsxtrshortpl}[2] [] {%
7644   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
7645 }

    Read in the final optional argument:
7646 \def\@Glsxtrshortpl#1#2[#3] {%

    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7647   \@glsxtr@record{#1}{#2}{glslink}%
7648   \glstoifexists{#2}%
7649   {%
7650     \glssetabbrvfmt{\glscategory{#2}}%
7651     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7652     \let\glsxtrifwasfirstuse\@secondoftwo
7653     \let\glsifplural\@firstoftwo
7654     \let\glscapscase\@secondofthree
7655     \let\glsinsert\@empty
7656     \def\glscustomtext{%
7657       \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrininsertinside#3\fi}%
7658       \ifglsxtrininsertinside\else#3\fi
7659     }%
7660     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7661   }%
7662   \glspostlinkhook
7663 }

```

\GLSxtrshortpl

```

7664 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
7665 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
7666   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
7667 }

    Read in the final optional argument:
7668 \def\@GLSxtrshortpl#1#2[#3] {%

    If the record option has been used, the information needs to be written to the aux file regard-
    less of whether the entry exists (unless indexing has been switched off).
7669   \@glsxtr@record{#1}{#2}{glslink}%
7670   \glstoifexists{#2}%
7671   {%

```

```

7672 \glsssetabbrvfmt{\glscategory{#2}}%
7673 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7674 \let\glxtrifwasfirstuse\@secondoftwo
7675 \let\gl@ifplural\@firstoftwo
7676 \let\glscapscase\@thirdofthree
7677 \let\glinsert\@empty
7678 \def\glscustomtext{%
7679 \mfirstucMakeUppercase
7680 {\glabbrvfont{\glaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
7681 \ifglxtrininsertinside\else#3\fi
7682 }%
7683 }%
7684 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7685 }%
7686 \glspostlinkhook
7687 }

```

Plural long forms:

\glxtrlongpl

```

7688 \newrobustcmd*{\glxtrlongpl}{\@gl@hyp@opt\@ns@glxtrlongpl}
  Define the un-starred form. Need to determine if there is a final optional argument
7689 \newcommand*{\ns@glxtrlongpl}[2] [] {%
7690 \new@ifnextchar[{\@glxtrlongpl{#1}{#2}}{\@glxtrlongpl{#1}{#2} []}%
7691 }

```

Read in the final optional argument:

```

7692 \def\@glxtrlongpl#1#2[#3] {%

```

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

```

7693 \@glxtr@record{#1}{#2}{glslink}%
7694 \glsoifexists{#2}%
7695 {%
7696 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
7697 \let\glxtrifwasfirstuse\@secondoftwo
7698 \let\gl@ifplural\@firstoftwo
7699 \let\glscapscase\@firstofthree
7700 \let\glinsert\@empty
7701 \def\glscustomtext{%
7702 \glslongfont{\glaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
7703 \ifglxtrininsertinside\else#3\fi
7704 }%
7705 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7706 }%
7707 \glspostlinkhook
7708 }

```

\Glsxtrlongpl

```

7709 \newrobustcmd*{\Glsxtrlongpl}{\@gl@hyp@opt\@ns@Glsxtrlongpl}

```


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

```
7710 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
7711   \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
7712 }
```

Read in the final optional argument:

```
7713 \def\@Glsxtrlongpl#1#2[#3] {%
```

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

```
7714   \@glsxtr@record{#1}{#2}{glslink}%
7715   \glsdoifexists{#2}%
7716   {%
7717     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7718     \let\glsxtrifwasfirstuse\@secondoftwo
7719     \let\glsifplural\@firstoftwo
7720     \let\glscapscase\@secondofthree
7721     \let\glsinsert\@empty
7722     \def\glscustomtext{%
7723       \glslongfont{\Glsaccesslongpl{#2}\ifglsxtrininsertinside#3\fi}%
7724       \ifglsxtrininsertinside\else#3\fi
7725     }%
7726     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7727   }%
7728   \glspostlinkhook
7729 }
```

\Glsxtrlongpl

```
7730 \newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\@ns@Glsxtrlongpl}
```

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

```
7731 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
7732   \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
7733 }
```

Read in the final optional argument:

```
7734 \def\@Glsxtrlongpl#1#2[#3] {%
```

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

```
7735   \@glsxtr@record{#1}{#2}{glslink}%
7736   \glsdoifexists{#2}%
7737   {%
7738     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
7739     \let\glsxtrifwasfirstuse\@secondoftwo
7740     \let\glsifplural\@firstoftwo
7741     \let\glscapscase\@thirdofthree
7742     \let\glsinsert\@empty
7743     \def\glscustomtext{%
7744       \mfirstucMakeUppercase
7745       {\glslongfont{\Glsaccesslongpl{#2}\ifglsxtrininsertinside#3\fi}%

```

```

7746      \ifglxtrinsertinside\else#3\fi
7747      }%
7748      }%
7749      \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
7750      }%
7751      \glspostlinkhook
7752 }

```

\glsetabbrvfmt Set the current format for the given category (or the abbreviation category if unset).

```

7753 \newcommand*{\glsetabbrvfmt}[1]{%
7754   \ifcsdef{@glsabbrv@current@#1}%
7755   {\glxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
7756   {\glxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
7757 }

```

glsuseabbrvfont Provide a way to use the abbreviation font for a given category for arbitrary text.

```

7758 \newrobustcmd*{\gluseabbrvfont}[2]{\glsetabbrvfmt{#2}\glsabbrvfont{#1}}

```

\glsuselongfont Provide a way to use the long font for a given category for arbitrary text.

```

7759 \newrobustcmd*{\glsuselongfont}[2]{\glsetabbrvfmt{#2}\glslongfont{#1}}

```

sxtrgenabbrvfmt Similar to \glsgenacfmt, but for abbreviations.

```

7760 \newcommand*{\glsxtrgenabbrvfmt}{%
7761   \ifdefempty\glscustomtext
7762   {%
7763     \ifglsused\glslabel
7764     {%

```

Subsequent use:

```

7765     \glsifplural
7766     {%

```

Subsequent plural form:

```

7767     \glscapscase
7768     {%

```

Subsequent plural form, don't adjust case:

```

7769     \glxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7770     }%
7771     {%

```

Subsequent plural form, make first letter upper case:

```

7772     \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
7773     }%
7774     {%

```

Subsequent plural form, all caps:

```

7775     \mfirstucMakeUppercase
7776     {\glxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
7777     }%

```

7778 }%
7779 {%

Subsequent singular form

7780 \glscapscase
7781 {%

Subsequent singular form, don't adjust case:

7782 \glxtrsubsequentfmt{\glslabel}{\glsinsert}%
7783 }%
7784 {%

Subsequent singular form, make first letter upper case:

7785 \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
7786 }%
7787 {%

Subsequent singular form, all caps:

7788 \mfirstucMakeUppercase
7789 {\glxtrsubsequentfmt{\glslabel}{\glsinsert}}%
7790 }%
7791 }%
7792 }%
7793 {%

First use:

7794 \glsifplural
7795 {%

First use plural form:

7796 \glscapscase
7797 {%

First use plural form, don't adjust case:

7798 \glxtrfullplformat{\glslabel}{\glsinsert}%
7799 }%
7800 {%

First use plural form, make first letter upper case:

7801 \Glsxtrfullplformat{\glslabel}{\glsinsert}%
7802 }%
7803 {%

First use plural form, all caps:

7804 \mfirstucMakeUppercase
7805 {\glxtrfullplformat{\glslabel}{\glsinsert}}%
7806 }%
7807 }%
7808 {%

First use singular form

7809 \glscapscase
7810 {%

First use singular form, don't adjust case:

```
7811      \glxtrfullformat{\glslabel}{\glsinsert}%
7812      }%
7813      {%
```

First use singular form, make first letter upper case:

```
7814      \Glsxtrfullformat{\glslabel}{\glsinsert}%
7815      }%
7816      {%
```

First use singular form, all caps:

```
7817      \mfirstucMakeUppercase
7818      {\glxtrfullformat{\glslabel}{\glsinsert}}%
7819      }%
7820      }%
7821      }%
7822      }%
7823      {%
```

User supplied text.

```
7824      \glscustomtext
7825      }%
7826 }
```

trsubsequentfmt Subsequent use format (singular no case change).

```
7827 \newcommand*{\glxtrsubsequentfmt}[2]{%
7828   \glsabbrvfont{\glsaccessshort{#1}\ifglxtrininsertinside #2\fi}%
7829   \ifglxtrininsertinside \else#2\fi
7830 }
7831 \let\glxtrdefaultsubsequentfmt\glxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural no case change).

```
7832 \newcommand*{\glxtrsubsequentplfmt}[2]{%
7833   \glsabbrvfont{\glsaccessshortpl{#1}\ifglxtrininsertinside #2\fi}%
7834   \ifglxtrininsertinside \else#2\fi
7835 }
7836 \let\glxtrdefaultsubsequentplfmt\glxtrsubsequentplfmt
```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).

```
7837 \newcommand*{\Glsxtrsubsequentfmt}[2]{%
7838   \glsabbrvfont{\Glsaccessshort{#1}\ifglxtrininsertinside #2\fi}%
7839   \ifglxtrininsertinside \else#2\fi
7840 }
7841 \let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural, first letter uppercase).

```
7842 \newcommand*{\Glsxtrsubsequentplfmt}[2]{%
7843   \glsabbrvfont{\Glsaccessshortpl{#1}\ifglxtrininsertinside #2\fi}%
7844   \ifglxtrininsertinside \else#2\fi
7845 }
7846 \let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt
```

1.7.1 Abbreviation Styles Setup

abbreviationstyle

```

7847 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
7848   \ifcsundef{@glsabbrv@dispstyle@setup@#2}%
7849   {%
7850     \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
7851   }%
7852   {%

```

Have abbreviations already been defined for this category?

```

7853   \ifcsstring{@glsabbrv@current@#1}{#2}%
7854   {%

```

Style already set.

```

7855   }%
7856   {%
7857     \def\@glsxtr@dostylewarn{%
7858       \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
7859       {%
7860         \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
7861           style has been switched \MessageBreak
7862           for category ‘#1’, \MessageBreak
7863           but there have already been entries \MessageBreak
7864           defined for this category. Unwanted \MessageBreak
7865           side-effects may result}}%
7866         \@endfortrue
7867       }%
7868       \@glsxtr@dostylewarn

```

Set up the style for the given category.

```

7869     \csdef{@glsabbrv@current@#1}{#2}%
7870     \edef\glscategorylabel{#1}%
7871     \glsxtr@applyabbrvstyle{#2}%
7872   }%
7873 }%
7874 }

```

applyabbrvstyle Apply the abbreviation style without existence check.

```

7875 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
7876   \csuse{@glsabbrv@dispstyle@setup@#1}%
7877   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7878 }

```

r@applyabbrvfmt Only apply the style formats.

```

7879 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
7880   \csuse{@glsabbrv@dispstyle@fmts@#1}%
7881 }

```

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.

```

7882 \newcommand*{\newabbreviationstyle}[3]{%
7883   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
7884   {%
7885     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
7886       defined}{}}%
7887   }%
7888   {%
7889     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

7890     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7891     #2}%
7892     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

7893     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7894     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7895     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7896     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

Reset `\glsxtrsubsequentfmt` etc in case a style changes this.

```

7897     \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
7898     \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
7899     \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
7900     \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
7901     #3}%
7902   }%
7903 }

```

breivationstyle

```

7904 \newcommand*{\renewabbreviationstyle}[3]{%
7905   \ifcsundef{@glsabbrv@dispstyle@setup@#1}
7906   {%
7907     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
7908   }%
7909   {%
7910     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```

7911     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
7912     #2}%
7913     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```

7914     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
7915     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
7916     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
7917     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%

```

```

7918     #3}%
7919   }%
7920 }

```

abbreviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```

7921 \newcommand*{\letabbreviationstyle}[2]{%
7922   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
7923   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7924 }

```

deprecated@abbrstyle

```
\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}
```

Define a synonym for a deprecated abbreviation style.

```

7925 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
7926   \csdef{@glsabbrv@dispstyle@setup@#1}{%
7927     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
7928     \csuse{@glsabbrv@dispstyle@setup@#2}%
7929   }%
7930   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
7931 }

```

deprecatedAbbrStyle Generate warning for deprecated style use.

```

7932 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
7933   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
7934   use ‘#2’ instead}%
7935 }

```

useAbbrStyleSetup

```

7936 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
7937   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
7938   {%
7939     \PackageError{glossaries-extra}%
7940     {Unknown abbreviation style definitions ‘#1’}{}%
7941   }%
7942   {%
7943     \csname @glsabbrv@dispstyle@setup@#1\endcsname
7944   }%
7945 }

```

useAbbrStyleFmt

```

7946 \newcommand*{\GlsXtrUseAbbrStyleFmt}[1]{%
7947   \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
7948   {%
7949     \PackageError{glossaries-extra}%

```

```

7950     {Unknown abbreviation style formats ‘#1’}{}%
7951 }%
7952 {%
7953     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
7954 }%
7955 }

```

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.

```

7956 \newif\ifglsxtrinsertinside
7957 \glsxtrinsertinsidefalse

```

`trlongshortname`

```

7958 \newcommand*{\glsxtrlongshortname}{%
7959     \protect\glsabbrvfont{\the\glsshorttok}%
7960 }

```

`long-short`

```

7961 \newabbreviationstyle{long-short}%
7962 {%

```

Set accessibility attributes if enabled.

```

7963     \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

7964     \renewcommand*{\CustomAbbreviationFields}{%
7965         name={\glsxtrlongshortname},
7966         sort={\the\glsshorttok},
7967         first={\protect\glsfirstlongfont{\the\glslongtok}%
7968             \protect\glsxtrfullsep{\the\glslabeltok}%
7969             \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
7970         firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
7971             \protect\glsxtrfullsep{\the\glslabeltok}%
7972             \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
7973         plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
7974         text={\protect\glsabbrvfont{\the\glsshorttok}},%
7975         description={\the\glslongtok}}%

```


Unset the regular attribute if it has been set.

```

7976 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
7977   \glshasattribute{\the\glslabeltok}{regular}%
7978   {%
7979     \glsselattribute{\the\glslabeltok}{regular}{false}%
7980   }%
7981   {}%
7982 }%
7983 }%
7984 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

7985 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
7986 \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
7987 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
7988 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
7989 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

7990 \renewcommand*{\glsxtrfullformat}[2]{%
7991   \glsfirstlongfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
7992   \ifglsxtrininsertinside\else##2\fi
7993   \glsxtrfullsep{##1}%
7994   \glsxtrparen{\glsfirstabbrvfont{\glssaccessshort{##1}}}%
7995 }%
7996 \renewcommand*{\glsxtrfullplformat}[2]{%
7997   \glsfirstlongfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
7998   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
7999   \glsxtrparen{\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%
8000 }%
8001 \renewcommand*{\Glsxtrfullformat}[2]{%
8002   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8003   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8004   \glsxtrparen{\glsfirstabbrvfont{\glssaccessshort{##1}}}%
8005 }%
8006 \renewcommand*{\Glsxtrfullplformat}[2]{%
8007   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8008   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8009   \glsxtrparen{\glsfirstabbrvfont{\glssaccessshortpl{##1}}}%
8010 }%
8011 }

```

Set this as the default style for general abbreviations:

```

8012 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

8013 \newcommand*{\glsxtrlongshortdescsort}{%
8014   \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
8015 }

```

ngshortdescname

```
8016 \newcommand*{\glxtrlongshortdescname}{%
8017   \protect\glslongfont{\the\glslongtok}
8018   \glxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
8019 }
```

long-short-desc User supplies description. The long form is included in the name.

```
8020 \newabbreviationstyle{long-short-desc}%
8021 {%
  Set accessibility attributes if enabled.
8022   \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
  Setup the default fields.
8023   \renewcommand*{\CustomAbbreviationFields}{%
8024     name={\glxtrlongshortdescname},
8025     sort={\glxtrlongshortdescsort},%
8026     first={\protect\glsfirstlongfont{\the\glslongtok}%
8027       \protect\glxtrfullsep{\the\glslabeltok}%
8028       \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
8029     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
8030       \protect\glxtrfullsep{\the\glslabeltok}%
8031       \glxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
  The text key should only have the short form.
8032   text={\protect\glsabbrvfont{\the\glsshorttok}},%
8033   plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
8034 }%
```

Unset the regular attribute if it has been set.

```
8035 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8036   \glshasattribute{\the\glslabeltok}{regular}%
8037   {%
8038     \glissetattribute{\the\glslabeltok}{regular}{false}%
8039   }%
8040   {}%
8041 }%
8042 }%
8043 {%
8044   \GlsXtrUseAbbrStyleFmts{long-short}%
8045 }
```

trshortlongname

```
8046 \newcommand*{\glxtrshortlongname}{%
8047   \protect\glsabbrvfont{\the\glsshorttok}%
8048 }
```

short-long Short form followed by long form in parenthesis on first use.

```
8049 \newabbreviationstyle{short-long}%
8050 {%
```

Set accessibility attributes if enabled.

```
8051 \glxstrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
8052 \renewcommand*{\CustomAbbreviationFields}{%
8053   name={\glxstrshortlongname},
8054   sort={\the\glsshorttok},
8055   description={\the\glslongtok},%
8056   first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
8057     \protect\glxstrfullsep{\the\glslabeltok}%
8058     \glxstrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
8059   firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
8060     \protect\glxstrfullsep{\the\glslabeltok}%
8061     \glxstrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%

8062   text={\protect\glsabbrvfont{\the\glsshorttok}},%
8063   plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
8064 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8065   \glshasattribute{\the\glslabeltok}{regular}%
8066   {%
8067     \glissetattribute{\the\glslabeltok}{regular}{false}%
8068   }%
8069   {}%
8070 }%
8071 }%
8072 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
8073 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8074 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
8075 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
8076 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8077 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8078 \renewcommand*{\glxstrfullformat}[2]{%
8079   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
8080   \ifglxtrininsertinside\else##2\fi
8081   \glxstrfullsep{##1}%
8082   \glxstrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
8083 }%
8084 \renewcommand*{\glxstrfullplformat}[2]{%
8085   \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
8086   \ifglxtrininsertinside\else##2\fi
8087   \glxstrfullsep{##1}%
8088   \glxstrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
8089 }%
8090 \renewcommand*{\Glsxtrfullformat}[2]{%
8091   \glsfirstabbrvfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%

```

```

8092 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8093 \glxtrparen{\glsfirstlongfont{\glssaccesslong{##1}}}%
8094 }%
8095 \renewcommand*{\Glsxtrfullplformat}[2]{%
8096 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8097 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8098 \glxtrparen{\glsfirstlongfont{\glssaccesslongpl{##1}}}%
8099 }%
8100 }

```

ortlongdescsort

```
8101 \newcommand*{\glxtrshortlongdescsort}{\the\glssshorttok}
```

ortlongdescname

```

8102 \newcommand*{\glxtrshortlongdescname}{%
8103 \protect\glssabbrvfont{\the\glssshorttok}
8104 \glxtrparen{\protect\glslongfont{\the\glslongtok}}}%
8105 }

```

short-long-desc User supplies description. The long form is included in the name.

```

8106 \newabbreviationstyle{short-long-desc}%
8107 {%

```

Set accessibility attributes if enabled.

```
8108 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```

8109 \renewcommand*{\CustomAbbreviationFields}{%
8110 name={\glxtrshortlongdescname},
8111 sort={\glxtrshortlongdescsort},
8112 first={\protect\glsfirstabbrvfont{\the\glssshorttok}%
8113 \protect\glxtrfullsep{\the\glslabeltok}%
8114 \glxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
8115 firstplural={\protect\glsfirstabbrvfont{\the\glssshortpltok}%
8116 \protect\glxtrfullsep{\the\glslabeltok}%
8117 \glxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
8118 text={\protect\glssabbrvfont{\the\glssshorttok}},%
8119 plural={\protect\glssabbrvfont{\the\glssshortpltok}}}%
8120 }%

```

Unset the regular attribute if it has been set.

```

8121 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8122 \glshasattribute{\the\glslabeltok}{regular}%
8123 {%
8124 \glissetattribute{\the\glslabeltok}{regular}{false}%
8125 }%
8126 {}%
8127 }%

```

```

8128 }%
8129 {%
8130   \GlsXtrUseAbbrStyleFmts{short-long}%
8131 }

```

`ongfootnotefont` Only used by the “footnote” styles.

```

8132 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%

```

`ongfootnotefont` Only used by the “footnote” styles.

```

8133 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%

```

`trabbrvfootnote`

`\glsxtrabbrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument `<long>` includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```

8134 \newcommand*{\glsxtrabbrvfootnote}[2]{\footnote{#2}}

```

`xtrfootnotename`

```

8135 \newcommand*{\glsxtrfootnotename}{%
8136   \protect\glsabbrvfont{\the\glsshorttok}%
8137 }

```

`footnote` Short form followed by long form in footnote on first use.

```

8138 \newabbreviationstyle{footnote}%
8139 {%

```

Set accessibility attributes if enabled. (Add `firstshortaccess` since long form is hidden in a footnote on first use.)

```

8140   \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

8141   \renewcommand*{\CustomAbbreviationFields}{%
8142     name={\glsxtrfootnotename},
8143     sort={\the\glsshorttok},
8144     description={\the\glslongtok},%

8145     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
8146       \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
8147       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8148     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
8149       \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
8150       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

```

```

8151     text={\protect\glsabbrvfont{\the\glsshorttok}},%
8152     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.

```

8153 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8154   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8155   \glsattribute{\the\glslabeltok}{regular}%
8156   {%
8157     \glssetattribute{\the\glslabeltok}{regular}{false}%
8158   }%
8159   {}%
8160 }%
8161 }%
8162 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8163 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
8164 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
8165 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
8166 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
8167 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

8168 \renewcommand*{\glsxtrfullformat}[2]{%
8169   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8170   \ifglsxtrininsertinside\else##2\fi
8171   \protect\glsxtrabbrvfootnote{##1}%
8172   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8173 }%
8174 \renewcommand*{\glsxtrfullplformat}[2]{%
8175   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8176   \ifglsxtrininsertinside\else##2\fi
8177   \protect\glsxtrabbrvfootnote{##1}%
8178   {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8179 }%
8180 \renewcommand*{\Glsxtrfullformat}[2]{%
8181   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
8182   \ifglsxtrininsertinside\else##2\fi
8183   \protect\glsxtrabbrvfootnote{##1}%
8184   {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
8185 }%
8186 \renewcommand*{\Glsxtrfullplformat}[2]{%
8187   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
8188   \ifglsxtrininsertinside\else##2\fi
8189   \protect\glsxtrabbrvfootnote{##1}%
8190   {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
8191 }%

```

The first use full form and the inline full form use the short (long) style.

```

8192 \renewcommand*{\glsxtrinlinefullformat}[2]{%

```

```

8193 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8194 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8195 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8196 }%
8197 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8198 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8199 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8200 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8201 }%
8202 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8203 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8204 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8205 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8206 }%
8207 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8208 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8209 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8210 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8211 }%
8212 }

```

short-footnote

```
8213 \letabbreviationstyle{short-footnote}{footnote}
```

ootnotedesname

```

8214 \newcommand*{\glxtrfootnotedesname}{%
8215 \protect\glsabbrvfont{\the\glsshorttok}%
8216 \protect\glxtrfullsep{\the\glslabeltok}%
8217 \protect\glxtrparen{\protect\glslongfont{\the\glslongtok}}}%
8218 }

```

ootnotedesort

```
8219 \newcommand*{\glxtrfootnotedesort}{\the\glsshorttok}
```

t-footnote-desc Like short-footnote but with user supplied description.

```

8220 \newabbreviationstyle{short-footnote-desc}%
8221 {%

```

Set accessibility attributes if enabled

```
8222 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```

8223 \renewcommand*{\CustomAbbreviationFields}{%
8224 name={\glxtrfootnotedesname},
8225 sort={\glxtrfootnotedesort},
8226 first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
8227 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
8228 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
8229 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
8230 \protect\glxtrabbrvfootnote{\the\glslabeltok}%

```

```

8231     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
8232     text={\protect\glsabbrvfont{\the\glsshorttok}},%
8233     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.

```

8234 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8235   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
8236   \glsattribute{\the\glslabeltok}{regular}%
8237   {%
8238     \glssetattribute{\the\glslabeltok}{regular}{false}%
8239   }%
8240   {}}%
8241 }%
8242 }%
8243 {%
8244   \GlsXtrUseAbbrStyleFmts{footnote}%
8245 }

```

footnote-desc Synonym.

```

8246 \letabbreviationstyle{footnote-desc}{short-footnote-desc}

```

postfootnote Similar to footnote 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.

```

8247 \newabbreviationstyle{postfootnote}%
8248 {%

```

Set accessibility attributes if enabled. (Add `firstshortaccess` since long form is hidden in a footnote on first use.)

```

8249   \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

8250 \renewcommand*{\CustomAbbreviationFields}{%
8251   name={\glsxtrfootnotename},
8252   sort={\the\glsshorttok},
8253   description={\the\glslongtok},%
8254   first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
8255   firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%

8256   text={\protect\glsabbrvfont{\the\glsshorttok}},%
8257   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.

```

8258 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8259   \csdef{glsxtrpostlink\glscategorylabel}{%
8260     \glsxtrifwasfirstuse
8261     {%

```


Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8262      \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
8263      {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
8264      }%
8265      {%
8266      }%
8267      \glshasattribute{\the\glslabeltok}{regular}%
8268      {%
8269      \glissetattribute{\the\glslabeltok}{regular}{false}%
8270      }%
8271      {%}%
8272      }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

8273  \renewcommand*{\glxtrsetupfulldefs}{%
8274  \let\glxtrifwasfirstuse\@secondoftwo
8275  }%
8276 }%
8277 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8278  \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
8279  \renewcommand*{\glxtrabbrvfont}[1]{\glxtrabbrvdefaultfont{##1}}%
8280  \renewcommand*{\glxtrfirstabbrvfont}[1]{\glxtrfirstabbrvdefaultfont{##1}}%
8281  \renewcommand*{\glxtrfirstlongfont}[1]{\glxtrfirstlongfootnotefont{##1}}%
8282  \renewcommand*{\glxtrlongfont}[1]{\glxtrlongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

8283  \renewcommand*{\glxtrfullformat}[2]{%
8284  \glxtrfirstabbrvfont{\glxtraccessshort{##1}\ifglxtrinsertinside##2\fi}%
8285  \ifglxtrinsertinside\else##2\fi
8286  }%
8287  \renewcommand*{\glxtrfullplformat}[2]{%
8288  \glxtrfirstabbrvfont{\glxtraccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8289  \ifglxtrinsertinside\else##2\fi
8290  }%
8291  \renewcommand*{\Glsxtrfullformat}[2]{%
8292  \glxtrfirstabbrvfont{\Glsxtraccessshort{##1}\ifglxtrinsertinside##2\fi}%
8293  \ifglxtrinsertinside\else##2\fi
8294  }%
8295  \renewcommand*{\Glsxtrfullplformat}[2]{%
8296  \glxtrfirstabbrvfont{\Glsxtraccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8297  \ifglxtrinsertinside\else##2\fi
8298  }%

```

The first use full form and the inline full form use the short (long) style.

```

8299  \renewcommand*{\glxtrininlinefullformat}[2]{%
8300  \glxtrfirstabbrvfont{\glxtraccessshort{##1}\ifglxtrinsertinside##2\fi}%
8301  \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

8302 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8303 }%
8304 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8305 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
8306 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8307 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8308 }%
8309 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8310 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
8311 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8312 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
8313 }%
8314 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8315 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
8316 \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8317 \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
8318 }%
8319 }

```

rt-postfootnote

```
8320 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

stfootnote-desc Like short-postfootnote but with user supplied description.

```

8321 \newabbreviationstyle{short-postfootnote-desc}%
8322 {%

```

Set accessibility attributes if enabled.

```
8323 \glsxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```

8324 \renewcommand*{\CustomAbbreviationFields}{%
8325 name={\glsxtrfootnotedesname},
8326 sort={\glsxtrfootnotedesort},
8327 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
8328 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
8329 text={\protect\glsabbrvfont{\the\glsshorttok}},%
8330 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.

```

8331 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8332 \csdef{glsxtrpostlink\glscategorylabel}{%
8333 \glsxtrifwasfirstuse
8334 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

8335 \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
8336 {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}%

```

```

8337     }%
8338     {}%
8339     }%
8340     \glshasattribute{\the\glslabeltok}{regular}%
8341     {%
8342         \glissetattribute{\the\glslabeltok}{regular}{false}%
8343     }%
8344     {}%
8345     }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

8346 \renewcommand*{\glxtrsetupfulldefs}{%
8347     \let\glxtrifwasfirstuse\@secondoftwo
8348 }%
8349 }%
8350 {%
8351     \GlsXtrUseAbbrStyleFmts{postfootnote}%
8352 }

```

stfootnote-desc

```

8353 \letabbreviationstyle{postfootnote-desc}{short-postfootnote-desc}

```

shortnolongname

```

8354 \newcommand*{\glxtrshortnolongname}{%
8355     \protect\glsabbrvfont{\the\glsshorttok}%
8356 }

```

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.

```

8357 \newabbreviationstyle{short}%
8358 {%

```

Set accessibility attributes if enabled.

```

8359 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

8360 \renewcommand*{\CustomAbbreviationFields}{%
8361     name={\glxtrshortnolongname},
8362     sort={\the\glsshorttok},
8363     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
8364     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
8365     text={\protect\glsabbrvfont{\the\glsshorttok}},
8366     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
8367     description={\the\glslongtok}}%
8368 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8369     \glissetattribute{\the\glslabeltok}{regular}{true}}%
8370 }%

```

8371 {%

In case the user wants to mix and match font styles, these are redefined here.

```
8372 \renewcommand*\abbrvpluralsuffix{\glstrabbrvpluralsuffix}%
8373 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
8374 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
8375 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8376 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
8377 \renewcommand*\glstrinlinefullformat[2]{%
8378   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
8379   \ifglstrinsertinside##2\fi}%
8380 \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
8381 \glstrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
8382 }%
8383 \renewcommand*\glstrinlinefullplformat[2]{%
8384   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
8385   \ifglstrinsertinside##2\fi}%
8386 \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
8387 \glstrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
8388 }%
8389 \renewcommand*\Glsxtrinlinefullformat[2]{%
8390   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
8391   \ifglstrinsertinside##2\fi}%
8392 \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
8393 \glstrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
8394 }%
8395 \renewcommand*\Glsxtrinlinefullplformat[2]{%
8396   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
8397   \ifglstrinsertinside##2\fi}%
8398 \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
8399 \glstrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
8400 }%
```

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.

```
8401 \renewcommand*\glstrfullformat[2]{%
8402   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglstrinsertinside##2\fi}%
8403 \ifglstrinsertinside\else##2\fi
8404 }%
8405 \renewcommand*\glstrfullplformat[2]{%
8406   \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglstrinsertinside##2\fi}%
8407 \ifglstrinsertinside\else##2\fi
8408 }%
8409 \renewcommand*\Glsxtrfullformat[2]{%
8410   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglstrinsertinside##2\fi}%
8411 \ifglstrinsertinside\else##2\fi
8412 }%
8413 \renewcommand*\Glsxtrfullplformat[2]{%
8414   \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglstrinsertinside##2\fi}%
8415 }
```

```

8415 \ifglxtrinsertinside\else##2\fi
8416 }%
8417 }

```

Set this as the default style for acronyms:

```
8418 \setabbreviationstyle[acronym]{short}
```

short-nolong

```
8419 \letabbreviationstyle{short-nolong}{short}
```

rt-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```

8420 \newabbreviationstyle{short-nolong-noreg}%
8421 {%
8422 \GlsXtrUseAbbrStyleSetup{short-nolong}%

```

Unset the regular attribute if it has been set.

```

8423 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8424 \glshasattribute{\the\glslabeltok}{regular}%
8425 {%
8426 \glissetattribute{\the\glslabeltok}{regular}{false}%
8427 }%
8428 {}%
8429 }%
8430 }%
8431 {%
8432 \GlsXtrUseAbbrStyleFmts{short-nolong}%
8433 }

```

trshortdescname

```

8434 \newcommand*{\glxtrshortdescname}{%
8435 \protect\glsabbrvfont{\the\glsshorttok}%
8436 \protect\glxtrfullsep{\the\glslabeltok}%
8437 \protect\glxtrparen{\protect\glslongfont{\the\glslongtok}}%
8438 }

```

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.

```

8439 \newabbreviationstyle{short-desc}%
8440 {%

```

Set accessibility attributes if enabled.

```
8441 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```

8442 \renewcommand*{\CustomAbbreviationFields}{%
8443 name={\glxtrshortdescname},
8444 sort={\the\glsshorttok},
8445 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
8446 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},

```

```

8447   text={\protect\glsabbrvfont{\the\glsshorttok}},
8448   plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
8449 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8450   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8451 }%
8452 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8453 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
8454 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
8455 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
8456 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8457 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8458 \renewcommand*{\glsxtrinlinefullformat}[2]{%
8459   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8460   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8461   \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
8462 }%
8463 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8464   \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8465   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8466   \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
8467 }%
8468 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8469   \glsfirstabbrvfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8470   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8471   \glsxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
8472 }%
8473 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8474   \glsfirstabbrvfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8475   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8476   \glsxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
8477 }%

```

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.

```

8478 \renewcommand*{\glsxtrfullformat}[2]{%
8479   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8480   \ifglsxtrininsertinside\else##2\fi
8481 }%
8482 \renewcommand*{\glsxtrfullplformat}[2]{%
8483   \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8484   \ifglsxtrininsertinside\else##2\fi
8485 }%
8486 \renewcommand*{\Glsxtrfullformat}[2]{%
8487   \glsfirstabbrvfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8488   \ifglsxtrininsertinside\else##2\fi
8489 }%
8490 \renewcommand*{\Glsxtrfullplformat}[2]{%

```

```

8491 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8492 \ifglxtrinsertinside\else##2\fi
8493 }%
8494 }

```

ort-nolong-desc

```
8495 \letabbreviationstyle{short-nolong-desc}{short-desc}
```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```

8496 \newabbreviationstyle{short-nolong-desc-noreg}%
8497 {%
8498 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
      Unset the regular attribute if it has been set.
8499 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8500 \glshasattribute{\the\glslabeltok}{regular}%
8501 {%
8502 \glissetattribute{\the\glslabeltok}{regular}{false}%
8503 }%
8504 {}%
8505 }%
8506 }%
8507 {%
8508 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
8509 }

```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```

8510 \newabbreviationstyle{nolong-short}%
8511 {%
8512 \GlsXtrUseAbbrStyleSetup{short-nolong}%
8513 }%
8514 {%
8515 \GlsXtrUseAbbrStyleFmts{short-nolong}%
      The inline full form displays the long form followed by the short form in parentheses.
8516 \renewcommand*{\glxtrinlinefullformat}[2]{%
8517 \protect\glsfirstlongfont{\glsaccesslong{##1}%
8518 \ifglxtrinsertinside##2\fi}%
8519 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8520 \glxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8521 }%
8522 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8523 \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
8524 \ifglxtrinsertinside##2\fi}%
8525 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8526 \glxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8527 }%
8528 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8529 \protect\glsfirstlongfont{\glsaccesslong{##1}%

```

```

8530     \ifglxtrinsertinside##2\fi}%
8531     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8532     \glxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
8533 }%
8534 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8535     \protect\glsfirstlongfont{\Glsaccesslongpl{##1}%
8536         \ifglxtrinsertinside##2\fi}%
8537     \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8538     \glxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
8539 }%
8540 }

```

ong-short-noreg Like `nolong-short` but doesn't set the regular attribute.

```

8541 \newabbreviationstyle{nolong-short-noreg}%
8542 {%
8543     \GlsXtrUseAbbrStyleSetup{nolong-short}%
8544     \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8545         \glshasattribute{\the\glslabeltok}{regular}%
8546         {%
8547             \glissetattribute{\the\glslabeltok}{regular}{false}%
8548         }%
8549     }%
8550 }%
8551 }%
8552 {%
8553     \GlsXtrUseAbbrStyleFmts{nolong-short}%
8554 }

```

noshortdesname

```

8555 \newcommand*{\glxtrlongnoshortdesname}{%
8556     \protect\glslongfont{\the\glslongtok}%
8557 }

```

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. The accessibility attributes don't need setting here.

```

8558 \newabbreviationstyle{long-desc}%
8559 {%
8560     \renewcommand*{\CustomAbbreviationFields}{%
8561         name={\glxtrlongnoshortdesname},
8562         sort={\the\glslongtok},
8563         first={\protect\glsfirstlongfont{\the\glslongtok}},
8564         firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
8565         text={\glslongfont{\the\glslongtok}},
8566         plural={\glslongfont{\the\glslongpltok}}%
8567     }%

```



```

8568 \renewcommand*\GlsXtrPostNewAbbreviation}{%
8569   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8570 }%
8571 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

8572 \renewcommand*\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
8573 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
8574 \renewcommand*\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
8575 \renewcommand*\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8576 \renewcommand*\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

8577 \renewcommand*\glsxtrsubsequentfmt}[2]{%
8578   \glslongfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8579   \ifglsxtrininsertinside \else##2\fi
8580 }%
8581 \renewcommand*\glsxtrsubsequentplfmt}[2]{%
8582   \glslongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8583   \ifglsxtrininsertinside \else##2\fi
8584 }%
8585 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
8586   \glslongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
8587   \ifglsxtrininsertinside \else##2\fi
8588 }%
8589 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
8590   \glslongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
8591   \ifglsxtrininsertinside \else##2\fi
8592 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

8593 \renewcommand*\glsxtrinlinefullformat}[2]{%
8594   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8595   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8596   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8597 }%
8598 \renewcommand*\glsxtrinlinefullplformat}[2]{%
8599   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8600   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8601   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8602 }%
8603 \renewcommand*\Glsxtrinlinefullformat}[2]{%
8604   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
8605   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8606   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
8607 }%
8608 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
8609   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
8610   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8611   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
8612 }%

```

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.

```

8613 \renewcommand*{\glstrfullformat}[2]{%
8614   \glstrfirstlongfont{\glstraccesslong{##1}\ifglstrinsertinside##2\fi}%
8615   \ifglstrinsertinside\else##2\fi
8616 }%
8617 \renewcommand*{\glstrfullplformat}[2]{%
8618   \glstrfirstlongfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
8619   \ifglstrinsertinside\else##2\fi
8620 }%
8621 \renewcommand*{\Glsxtrfullformat}[2]{%
8622   \glstrfirstlongfont{\glstraccesslong{##1}\ifglstrinsertinside##2\fi}%
8623   \ifglstrinsertinside\else##2\fi
8624 }%
8625 \renewcommand*{\Glsxtrfullplformat}[2]{%
8626   \glstrfirstlongfont{\glstraccesslongpl{##1}\ifglstrinsertinside##2\fi}%
8627   \ifglstrinsertinside\else##2\fi
8628 }%
8629 }

```

long-noshort-desc Provide a synonym that matches similar styles.

```

8630 \letabbreviationstyle{long-noshort-desc}{long-desc}

```

short-desc-noreg Like long-noshort-desc but doesn't set the regular attribute.

```

8631 \newabbreviationstyle{long-noshort-desc-noreg}%
8632 {%
8633   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
8634   \UnsettheRegularAttribute
8635   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8636     \glshasattribute{\the\glslabeltok}{regular}%
8637     \glsssetattribute{\the\glslabeltok}{regular}{false}%
8638   }%
8639 }%
8640 }%
8641 }%
8642 {%
8643   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
8644 }

```

longnoshortname

```

8645 \newcommand*{\glstrlongnoshortname}{%
8646   \protect\glabbrvfont{\the\glssshorttok}%
8647 }

```

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.

```

8648 \newabbreviationstyle{long}%
8649 {%
    Set accessibility attributes if enabled.
8650   \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
    Setup the default fields.
8651   \renewcommand*{\CustomAbbreviationFields}{%
8652     name={\glxtrlongnoshortname},
8653     sort={\the\glsshorttok},
8654     first={\protect\glsfirstlongfont{\the\glslongtok}},
8655     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
8656     text={\glslongfont{\the\glslongtok}},
8657     plural={\glslongfont{\the\glslongpltok}},%
8658     description={\the\glslongtok}%
8659   }%
8660   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8661     \glsetattribute{\the\glslabeltok}{regular}{true}}%
8662 }%
8663 {%
8664   \GlsXtrUseAbbrStyleFmts{long-desc}%
8665 }

```

long-noshort Provide a synonym that matches similar styles.

```
8666 \letabbreviationstyle{long-noshort}{long}
```

g-noshort-noreg Like long-noshort but doesn't set the regular attribute.

```

8667 \newabbreviationstyle{long-noshort-noreg}%
8668 {%
8669   \GlsXtrUseAbbrStyleSetup{long-noshort}%
    Unset the regular attribute if it has been set.
8670   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8671     \glshasattribute{\the\glslabeltok}{regular}%
8672     {%
8673       \glsetattribute{\the\glslabeltok}{regular}{false}%
8674     }%
8675     {}%
8676   }%
8677 }%
8678 {%
8679   \GlsXtrUseAbbrStyleFmts{long-noshort}%
8680 }

```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

\glxtrscfont Maintained for backward-compatibility.

```
8681 \newcommand*{\glxtrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
8682 \newcommand*{\glsabbrvscfont}{\glsxtrscfont}
```

`\glsxtrfirstscfont` Maintained for backward-compatibility.

```
8683 \newcommand*{\glsxtrfirstscfont}[1]{\glsabbrvscfont{##1}}
```

`\glsfirstabbrvscfont` Added for consistent naming.

```
8684 \newcommand*{\glsfirstabbrvscfont}{\glsxtrfirstscfont}
```

and for the default short form suffix:

`\glsxtrscsuffix` `\protect` needs to come inside `\s` to avoid interfering with all caps.

```
8685 \newcommand*{\glsxtrscsuffix}{\protect\glstextup{\glsxtrabbrvpluralsuffix}}
```

`long-short-sc`

```
8686 \newabbreviationstyle{long-short-sc}%
8687 {%
```

Set accessibility attributes if enabled.

```
8688 \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
8689 \renewcommand*{\CustomAbbreviationFields}{%
8690   name={\glsxtrlongshortname},
8691   sort={\the\glsshorttok},
8692   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8693     \protect\glsxtrfullsep{\the\glslabeltok}%
8694     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
8695   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8696     \protect\glsxtrfullsep{\the\glslabeltok}%
8697     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8698   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8699   plural={\protect\glsabbrvscfont{\the\glsshortpltok}},%
8700   description={\the\glslongtok}}%
8701 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8702   \glshasattribute{\the\glslabeltok}{regular}%
8703   {%
8704     \glssetattribute{\the\glslabeltok}{regular}{false}%
8705   }%
8706   {}%
8707 }%
8708 }%
8709 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
8710 \renewcommand*{\abbrvpluralsuffix}{\glsxtrscsuffix}%
8711 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8712 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
```

Use the default long fonts.

```
8713 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8714 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
8715 \renewcommand*{\glsxtrfullformat}[2]{%
8716   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8717   \ifglsxtrinsertinside\else##2\fi
8718   \glsxtrfullsep{##1}%
8719   \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8720 }%
8721 \renewcommand*{\glsxtrfullplformat}[2]{%
8722   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8723   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8724   \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8725 }%
8726 \renewcommand*{\Glsxtrfullformat}[2]{%
8727   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
8728   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8729   \glsxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8730 }%
8731 \renewcommand*{\Glsxtrfullplformat}[2]{%
8732   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
8733   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
8734   \glsxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8735 }%
8736 }
```

g-short-sc-desc

```
8737 \newabbreviationstyle{long-short-sc-desc}%
8738 {%
```

Set accessibility attributes if enabled.

```
8739 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
8740 \renewcommand*{\CustomAbbreviationFields}{%
8741   name={\glsxtrlongshortdescname},
8742   sort={\glsxtrlongshortdescsort},%
8743   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
8744     \protect\glsxtrfullsep{\the\glslabeltok}%
8745     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
8746   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
8747     \protect\glsxtrfullsep{\the\glslabeltok}%
8748     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
8749   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8750   plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
8751 }%
```

Unset the regular attribute if it has been set.

```
8752 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```

8753 \glshasattribute{\the\glslabeltok}{regular}%
8754 {%
8755 \glissetattribute{\the\glslabeltok}{regular}{false}%
8756 }%
8757 {}%
8758 }%
8759 }%
8760 {%

```

As long-short-sc style:

```

8761 \GlsXtrUseAbbrStyleFmts{long-short-sc}%
8762 }

```

short-sc-long Now the short (long) version

```

8763 \newabbreviationstyle{short-sc-long}%
8764 {%

```

Set accessibility attributes if enabled.

```

8765 \glsextrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

8766 \renewcommand*{\CustomAbbreviationFields}{%
8767 name={\glsextrshortlongname},
8768 sort={\the\glsshorttok},
8769 description={\the\glslongtok},%
8770 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8771 \protect\glsextrfullsep{\the\glslabeltok}%
8772 \glsextrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8773 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8774 \protect\glsextrfullsep{\the\glslabeltok}%
8775 \glsextrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8776 text={\protect\glsabbrvscfont{\the\glsshorttok}},%
8777 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

8778 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8779 \glshasattribute{\the\glslabeltok}{regular}%
8780 {%
8781 \glissetattribute{\the\glslabeltok}{regular}{false}%
8782 }%
8783 {}%
8784 }%
8785 }%
8786 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8787 \renewcommand*{\abbrvpluralsuffix}{\glsextrscsuffix}%
8788 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8789 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8790 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8791 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline form are the same for this style.

```

8792 \renewcommand*{\glxtrfullformat}[2]{%
8793   \glsfirstabbrvscfont{\glaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8794   \ifglxtrinsertinside\else##2\fi
8795   \glxtrfullsep{##1}%
8796   \glxtrparen{\glsfirstlongdefaultfont{\glaccesslong{##1}}}%
8797 }%
8798 \renewcommand*{\glxtrfullplformat}[2]{%
8799   \glsfirstabbrvscfont{\glaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8800   \ifglxtrinsertinside\else##2\fi
8801   \glxtrfullsep{##1}%
8802   \glxtrparen{\glsfirstlongdefaultfont{\glaccesslongpl{##1}}}%
8803 }%
8804 \renewcommand*{\Glsxtrfullformat}[2]{%
8805   \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
8806   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8807   \glxtrparen{\glsfirstlongdefaultfont{\glaccesslong{##1}}}%
8808 }%
8809 \renewcommand*{\Glsxtrfullplformat}[2]{%
8810   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
8811   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8812   \glxtrparen{\glsfirstlongdefaultfont{\glaccesslongpl{##1}}}%
8813 }%
8814 }

```

rt-sc-long-desc As before but user provides description

```

8815 \newabbreviationstyle{short-sc-long-desc}%
8816 {%

```

Set accessibility attributes if enabled.

```

8817 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

8818 \renewcommand*{\CustomAbbreviationFields}{%
8819   name={\glxtrshortlongdescname},
8820   sort={\glxtrshortlongdescsort},
8821   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
8822     \protect\glxtrfullsep{\the\glslabeltok}%
8823     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
8824   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
8825     \protect\glxtrfullsep{\the\glslabeltok}%
8826     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
8827   text={\protect\glssabbrvscfont{\the\glsshorttok}},%
8828   plural={\protect\glssabbrvscfont{\the\glsshortpltok}}}%
8829 }%

```

Unset the regular attribute if it has been set.

```

8830 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8831   \glshasattribute{\the\glslabeltok}{regular}%
8832   {%

```

```

8833     \glssetattribute{\the\glslabeltok}{regular}{false}%
8834   }%
8835   {}%
8836 }%
8837 }%
8838 {%

```

As short-sc-long style:

```

8839   \GlsXtrUseAbbrStyleFmts{short-sc-long}%
8840 }

```

short-sc

```

8841 \newabbreviationstyle{short-sc}%
8842 {%

```

Set accessibility attributes if enabled.

```

8843   \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

8844   \renewcommand*{\CustomAbbreviationFields}{%
8845     name={\glsxtrshortnolongname},
8846     sort={\the\glsshorttok},
8847     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8848     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8849     text={\protect\glsabbrvscfont{\the\glsshorttok}},
8850     plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
8851     description={\the\glslongtok}}%
8852   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
8853     \glssetattribute{\the\glslabeltok}{regular}{true}}%
8854 }%
8855 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8856   \renewcommand*{\abbrvpluralsuffix}{\glsxtrscsuffix}%
8857   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
8858   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
8859   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
8860   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

8861   \renewcommand*{\glsxtrinlinefullformat}[2]{%
8862     \protect\glsfirstabbrvscfont{\glsaccessshort{##1}}%
8863     \ifglsxtrininsertinside##2\fi}%
8864     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8865     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8866   }%
8867   \renewcommand*{\glsxtrinlinefullplformat}[2]{%
8868     \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}%
8869     \ifglsxtrininsertinside##2\fi}%
8870     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8871     \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8872   }%

```



```

8873 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8874   \protect\glsfirstabbrvscfont{\Glsaccesssshort{##1}%
8875     \ifglxtrinsertinside##2\fi}%
8876   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8877   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
8878 }%
8879 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8880   \protect\glsfirstabbrvscfont{\Glsaccesssshortpl{##1}%
8881     \ifglxtrinsertinside##2\fi}%
8882   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8883   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
8884 }%

```

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.

```

8885 \renewcommand*{\glxtrfullformat}[2]{%
8886   \glsfirstabbrvscfont{\Glsaccesssshort{##1}\ifglxtrinsertinside##2\fi}%
8887   \ifglxtrinsertinside\else##2\fi
8888 }%
8889 \renewcommand*{\glxtrfullplformat}[2]{%
8890   \glsfirstabbrvscfont{\Glsaccesssshortpl{##1}\ifglxtrinsertinside##2\fi}%
8891   \ifglxtrinsertinside\else##2\fi
8892 }%
8893 \renewcommand*{\Glsxtrfullformat}[2]{%
8894   \glsfirstabbrvscfont{\Glsaccesssshort{##1}\ifglxtrinsertinside##2\fi}%
8895   \ifglxtrinsertinside\else##2\fi
8896 }%
8897 \renewcommand*{\Glsxtrfullplformat}[2]{%
8898   \glsfirstabbrvscfont{\Glsaccesssshortpl{##1}\ifglxtrinsertinside##2\fi}%
8899   \ifglxtrinsertinside\else##2\fi
8900 }%
8901 }

```

short-sc-nolong

```

8902 \letabbreviationstyle{short-sc-nolong}{short-sc}

```

short-sc-desc

```

8903 \newabbreviationstyle{short-sc-desc}%
8904 {%

```

Set accessibility attributes if enabled.

```

8905   \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```

Setup the default fields.

```

8906   \renewcommand*{\CustomAbbreviationFields}{%
8907     name={\glxtrshortdescname},
8908     sort={\the\glsshorttok},
8909     first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
8910     firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
8911     text={\protect\glsabbrvscfont{\the\glsshorttok}},

```

```

8912 plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
8913 \renewcommand*\GlsXtrPostNewAbbreviation{%
8914   \glssetattribute{\the\glslabeltok}{regular}{true}}%
8915 }%
8916 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

8917 \renewcommand*\abbrvpluralsuffix{\glsxtrscsuffix}%
8918 \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
8919 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
8920 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
8921 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

8922 \renewcommand*\glsxtrinlinefullformat[2]{%
8923   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8924   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8925   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8926 }%
8927 \renewcommand*\glsxtrinlinefullplformat[2]{%
8928   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8929   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8930   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8931 }%
8932 \renewcommand*\Glsxtrinlinefullformat[2]{%
8933   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8934   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8935   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
8936 }%
8937 \renewcommand*\Glsxtrinlinefullplformat[2]{%
8938   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8939   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
8940   \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
8941 }%

```

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.

```

8942 \renewcommand*\glsxtrfullformat[2]{%
8943   \glsfirstabbrvscfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8944   \ifglsxtrininsertinside\else##2\fi
8945 }%
8946 \renewcommand*\glsxtrfullplformat[2]{%
8947   \glsfirstabbrvscfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
8948   \ifglsxtrininsertinside\else##2\fi
8949 }%
8950 \renewcommand*\Glsxtrfullformat[2]{%
8951   \glsfirstabbrvscfont{\Glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
8952   \ifglsxtrininsertinside\else##2\fi
8953 }%
8954 \renewcommand*\Glsxtrfullplformat[2]{%
8955   \glsfirstabbrvscfont{\Glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%

```

```

8956     \ifglxtrinsertinside\else##2\fi
8957 }%
8958 }

```

-sc-nolong-desc

```

8959 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}

```

nolong-short-sc

```

8960 \newabbreviationstyle{nolong-short-sc}%
8961 {%
8962   \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
8963 }%
8964 {%
8965   \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

8966 \renewcommand*{\glxtrinlinefullformat}[2]{%
8967   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
8968     \ifglxtrinsertinside##2\fi}%
8969   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8970   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
8971 }%
8972 \renewcommand*{\glxtrinlinefullplformat}[2]{%
8973   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
8974     \ifglxtrinsertinside##2\fi}%
8975   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8976   \glxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
8977 }%
8978 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
8979   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
8980     \ifglxtrinsertinside##2\fi}%
8981   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8982   \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshort{##1}}}%
8983 }%
8984 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
8985   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
8986     \ifglxtrinsertinside##2\fi}%
8987   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
8988   \glxtrparen{\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
8989 }%
8990 }

```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glxtrshort`. No accessibility attributes needed here.

```

8991 \newabbreviationstyle{long-noshort-sc}%
8992 {%
8993   \renewcommand*{\CustomAbbreviationFields}{%
8994     name={\glxtrlongnoshortname},
8995     sort={\the\glsshorttok},

```

```

8996 first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
8997 firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
8998 text={\protect\glslongdefaultfont{\the\glslongtok}},
8999 plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9000 description={\the\glslongtok}%
9001 }%
9002 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9003   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9004 }%
9005 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

9006 \renewcommand*{\abbrvpluralsuffix}{\glsxtrscsuffix}%
9007 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
9008 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
9009 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9010 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9011 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
9012   \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9013   \ifglsxtrininsertinside \else##2\fi
9014 }%
9015 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
9016   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9017   \ifglsxtrininsertinside \else##2\fi
9018 }%
9019 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
9020   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9021   \ifglsxtrininsertinside \else##2\fi
9022 }%
9023 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9024   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9025   \ifglsxtrininsertinside \else##2\fi
9026 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9027 \renewcommand*{\glsxtrinlinefullformat}[2]{%
9028   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9029   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9030   \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
9031 }%
9032 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9033   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9034   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9035   \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
9036 }%
9037 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9038   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9039   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9040   \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%

```

```

9041 }%
9042 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9043   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9044   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9045   \glxtrparen{\protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}}}%
9046 }%

```

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.

```

9047 \renewcommand*{\glxtrfullformat}[2]{%
9048   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9049   \ifglxtrinsertinside\else##2\fi
9050 }%
9051 \renewcommand*{\glxtrfullplformat}[2]{%
9052   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9053   \ifglxtrinsertinside\else##2\fi
9054 }%
9055 \renewcommand*{\Glsxtrfullformat}[2]{%
9056   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9057   \ifglxtrinsertinside\else##2\fi
9058 }%
9059 \renewcommand*{\Glsxtrfullplformat}[2]{%
9060   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9061   \ifglxtrinsertinside\else##2\fi
9062 }%
9063 }

```

long-sc Backward compatibility:

```

9064 \@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`.

```

9065 \newabbreviationstyle{long-noshort-sc-desc}%
9066 {%
9067   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9068 }%
9069 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

9070 \renewcommand*{\abbrvpluralsuffix}{\glxtrscsuffix}%
9071 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
9072 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
9073 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9074 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9075 \renewcommand*{\glxtrsubsequentfmt}[2]{%
9076   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9077   \ifglxtrinsertinside \else##2\fi
9078 }%

```

```

9079 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
9080   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9081   \ifglxtrinsertinside \else##2\fi
9082 }%
9083 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9084   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9085   \ifglxtrinsertinside \else##2\fi
9086 }%
9087 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
9088   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9089   \ifglxtrinsertinside \else##2\fi
9090 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9091 \renewcommand*{\glxtrinlinefullformat}[2]{%
9092   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9093   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9094   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
9095 }%
9096 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9097   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9098   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9099   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
9100 }%
9101 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9102   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9103   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9104   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
9105 }%
9106 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9107   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9108   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9109   \glxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
9110 }%

```

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.

```

9111 \renewcommand*{\glxtrfullformat}[2]{%
9112   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9113   \ifglxtrinsertinside\else##2\fi
9114 }%
9115 \renewcommand*{\glxtrfullplformat}[2]{%
9116   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9117   \ifglxtrinsertinside\else##2\fi
9118 }%
9119 \renewcommand*{\Glsxtrfullformat}[2]{%
9120   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9121   \ifglxtrinsertinside\else##2\fi
9122 }%
9123 \renewcommand*{\Glsxtrfullplformat}[2]{%

```

```

9124 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9125 \ifglxtrinsertinside\else##2\fi
9126 }%
9127 }

```

long-desc-sc Backward compatibility:

```

9128 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}

```

ort-sc-footnote

```

9129 \newabbreviationstyle{short-sc-footnote}%
9130 {%

```

Set accessibility attributes if enabled.

```

9131 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

9132 \renewcommand*{\CustomAbbreviationFields}{%
9133   name={\glxtrfootnotename},
9134   sort={\the\glsshorttok},
9135   description={\the\glslongtok},%
9136   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
9137     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9138     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9139   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
9140     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9141     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9142   text={\protect\glsabbrvscfont{\the\glsshorttok}},%
9143   plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

9144 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9145   \glsetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9146   \glshasattribute{\the\glslabeltok}{regular}%
9147   {%
9148     \glsetattribute{\the\glslabeltok}{regular}{false}%
9149   }%
9150   {}%
9151 }%
9152 }%
9153 {%

```

Use smallcaps and adjust the plural suffix to revert to upright.

```

9154 \renewcommand*{\abbrvpluralsuffix}{\glxtrscsuffix}%
9155 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvscfont{##1}}%
9156 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvscfont{##1}}%
9157 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9158 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

9159 \renewcommand*{\glxtrfullformat}[2]{%

```

```

9160 \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9161 \ifglxtrinsertinside\else##2\fi
9162 \protect\glxtrabbrvfootnote{##1}%
9163 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9164 }%
9165 \renewcommand*{\glxtrfullplformat}[2]{%
9166 \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9167 \ifglxtrinsertinside\else##2\fi
9168 \protect\glxtrabbrvfootnote{##1}%
9169 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9170 }%
9171 \renewcommand*{\Glsxtrfullformat}[2]{%
9172 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9173 \ifglxtrinsertinside\else##2\fi
9174 \protect\glxtrabbrvfootnote{##1}%
9175 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9176 }%
9177 \renewcommand*{\Glsxtrfullplformat}[2]{%
9178 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9179 \ifglxtrinsertinside\else##2\fi
9180 \protect\glxtrabbrvfootnote{##1}%
9181 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9182 }%

```

The first use full form and the inline full form use the short (long) style.

```

9183 \renewcommand*{\glxtrininlinefullformat}[2]{%
9184 \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9185 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9186 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9187 }%
9188 \renewcommand*{\glxtrininlinefullplformat}[2]{%
9189 \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9190 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9191 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9192 }%
9193 \renewcommand*{\Glsxtrininlinefullformat}[2]{%
9194 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9195 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9196 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9197 }%
9198 \renewcommand*{\Glsxtrininlinefullplformat}[2]{%
9199 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9200 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9201 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9202 }%
9203 }

```

footnote-sc Backward compatibility:

```

9204 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}

```


c-footnote-desc Like short-sc-footnote but with user supplied description.

```
9205 \newabbreviationstyle{short-sc-footnote-desc}%  
9206 {%
```

Set accessibility attributes if enabled.

```
9207 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
9208 \renewcommand*{\CustomAbbreviationFields}{%  
9209   name={\glxtrfootnotedesname},  
9210   sort={\glxtrfootnotedesort},  
9211   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%  
9212     \protect\glxtrabbrvfootnote{\the\glslabeltok}%  
9213     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%  
9214   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%  
9215     \protect\glxtrabbrvfootnote{\the\glslabeltok}%  
9216     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%  
9217   text={\protect\glsabbrvscfont{\the\glsshorttok}},%  
9218   plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
9219 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
9220   \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%  
9221   \glshasattribute{\the\glslabeltok}{regular}%  
9222   {%  
9223     \glsssetattribute{\the\glslabeltok}{regular}{false}%  
9224   }%  
9225   {}%  
9226 }%  
9227 }%  
9228 {%  
9229   \GlsXtrUseAbbrStyleFmts{short-sc-footnote}%  
9230 }
```

sc-postfootnote

```
9231 \newabbreviationstyle{short-sc-postfootnote}%  
9232 {%
```

Set accessibility attributes if enabled.

```
9233 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
9234 \renewcommand*{\CustomAbbreviationFields}{%  
9235   name={\glxtrfootnotename},  
9236   sort={\the\glsshorttok},  
9237   description={\the\glslongtok},%  
9238   first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%  
9239   firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%  
9240   text={\protect\glsabbrvscfont{\the\glsshorttok}},%  
9241   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.

```
9242 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9243   \csdef{glxstrpostlink\glscategorylabel}{%
9244     \glxtrifwasfirstuse
9245     {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
9246       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9247       {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
9248     }%
9249   }%
9250 }%
9251 \glshasattribute{\the\glslabeltok}{regular}%
9252 {%
9253   \glissetattribute{\the\glslabeltok}{regular}{false}%
9254   }%
9255   }%
9256 }%
```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```
9257 \renewcommand*{\glxtrsetupfulldefs}{%
9258   \let\glxtrifwasfirstuse\@secondoftwo
9259 }%
9260 }%
9261 {%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
9262 \renewcommand*{\abbrvpluralsuffix}{\glxtrscsuffix}%
9263 \renewcommand*{\glxabbrvfont}[1]{\glxabbrvscfont{##1}}%
9264 \renewcommand*{\glxfirstabbrvfont}[1]{\glxfirstabbrvscfont{##1}}%
9265 \renewcommand*{\glxfirstlongfont}[1]{\glxfirstlongfootnotefont{##1}}%
9266 \renewcommand*{\glxlongfont}[1]{\glxlongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
9267 \renewcommand*{\glxtrfullformat}[2]{%
9268   \glxfirstabbrvscfont{\glxaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9269   \ifglxtrinsertinside\else##2\fi
9270 }%
9271 \renewcommand*{\glxtrfullplformat}[2]{%
9272   \glxfirstabbrvscfont{\glxaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9273   \ifglxtrinsertinside\else##2\fi
9274 }%
9275 \renewcommand*{\Glsxtrfullformat}[2]{%
9276   \glxfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9277   \ifglxtrinsertinside\else##2\fi
9278 }%
9279 \renewcommand*{\Glsxtrfullplformat}[2]{%
```

```

9280 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9281 \ifglxtrinsertinside\else##2\fi
9282 }%

```

The first use full form and the inline full form use the short (long) style.

```

9283 \renewcommand*{\glxtrinlinefullformat}[2]{%
9284 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9285 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9286 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9287 }%
9288 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9289 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9290 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9291 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9292 }%
9293 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9294 \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9295 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9296 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
9297 }%
9298 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9299 \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9300 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9301 \glxtrparen{\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
9302 }%
9303 }

```

postfootnote-sc Backward compatibility:

```

9304 \@glxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}

```

stfootnote-desc Like short-sc-footnote but with user supplied description.

```

9305 \newabbreviationstyle{short-sc-postfootnote-desc}%
9306 {}

```

Set accessibility attributes if enabled.

```

9307 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```

Setup the default fields.

```

9308 \renewcommand*{\CustomAbbreviationFields}{%
9309 name={\glxtrfootnotedesname},
9310 sort={\glxtrfootnotedesort},
9311 first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
9312 firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
9313 text={\protect\glsabbrvscfont{\the\glsshorttok}},%
9314 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.

```

9315 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9316 \csdef{glxtrpostlink\glscategorylabel}{%
9317 \glxtrifwasfirstuse

```

```

9318      {%
Needs the specific font command here as the style may have been lost by the time the foot-
note occurs.
9319      \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
9320      {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
9321      }%
9322      {%
9323      }%
9324      \glshasattribute{\the\glslabeltok}{regular}%
9325      {%
9326      \glissetattribute{\the\glslabeltok}{regular}{false}%
9327      }%
9328      {%
9329      }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9330 \renewcommand*{\glxtrsetupfulldefs}{%
9331 \let\glxtrifwasfirstuse\@secondoftwo
9332 }%
9333 }%
9334 {%
9335 \GlsXtrUseAbbrStyleFmts{short-sc-postfootnote}%
9336 }

```

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.
9337 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}

\glsabbrvsmfont Added for consistent naming.
9338 \newcommand*{\glsabbrvsmfont}{\glxtrsmfont}

sxtrfirstsmfont Maintained for backward compatibility.
9339 \newcommand*{\glxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}

irstabbrvsmfont Added for consistent naming.
9340 \newcommand*{\glsfirstabbrvsmfont}{\glxtrfirstsmfont}

and for the default short form suffix:

\glxtrsmsuffix
9341 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}

```

long-short-sm

```
9342 \newabbreviationstyle{long-short-sm}%  
9343 {%
```

Set accessibility attributes if enabled.

```
9344 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
9345 \renewcommand*{\CustomAbbreviationFields}{%  
9346   name={\glxtrlongshortname},  
9347   sort={\the\glsshorttok},  
9348   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%  
9349     \protect\glxtrfullsep{\the\glslabeltok}%  
9350     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%  
9351   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%  
9352     \protect\glxtrfullsep{\the\glslabeltok}%  
9353     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%  
9354   text={\protect\glsabbrvsmfont{\the\glsshorttok}},%  
9355   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},%  
9356   description={\the\glslongtok}}%  
9357 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
9358   \glshasattribute{\the\glslabeltok}{regular}%  
9359   {%  
9360     \glissetattribute{\the\glslabeltok}{regular}{false}%  
9361   }%  
9362   {}%  
9363 }%  
9364 }%  
9365 {%
```

```
9366 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%  
9367 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%  
9368 \renewcommand*\abbrvpluralsuffix{\glsxtrsmsuffix}%
```

Use the default long fonts.

```
9369 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%  
9370 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9371 \renewcommand*\glxtrfullformat[2]{%  
9372   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%  
9373   \ifglxtrininsertinside\else##2\fi  
9374   \glxtrfullsep{##1}%  
9375   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshort{##1}}}%  
9376 }%  
9377 \renewcommand*\glxtrfullplformat[2]{%  
9378   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%  
9379   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%  
9380   \glxtrparen{\glsfirstabbrvsmfont{\glssaccessshortpl{##1}}}%  
9381 }%  
9382 \renewcommand*\Glsxtrfullformat[2]{%
```

```

9383 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9384 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9385 \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9386 }%
9387 \renewcommand*{\Glsxtrfullplformat}[2]{%
9388 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9389 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9390 \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9391 }%
9392 }

```

g-short-sm-desc

```

9393 \newabbreviationstyle{long-short-sm-desc}%
9394 {%
    Set accessibility attributes if enabled.
9395 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

9396 \renewcommand*{\CustomAbbreviationFields}{%
9397   name={\glxtrlongshortdescname},
9398   sort={\glxtrlongshortdescsort},%
9399   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
9400     \protect\glxtrfullsep{\the\glslabeltok}%
9401     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
9402   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
9403     \protect\glxtrfullsep{\the\glslabeltok}%
9404     \glxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
9405   text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
9406   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
9407 }%

```

Unset the regular attribute if it has been set.

```

9408 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9409 \glshasattribute{\the\glslabeltok}{regular}%
9410 {%
9411 \glissetattribute{\the\glslabeltok}{regular}{false}%
9412 }%
9413 {}%
9414 }%
9415 }%
9416 {%

```

As long-short-sm style:

```

9417 \GlsXtrUseAbbrStyleFmts{long-short-sm}%
9418 }

```

short-sm-long Now the short (long) version

```

9419 \newabbreviationstyle{short-sm-long}%
9420 {%

```

Set accessibility attributes if enabled.

```
9421 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
9422 \renewcommand*{\CustomAbbreviationFields}{%
9423   name={\glxtrshortlongname},
9424   sort={\the\glsshorttok},
9425   description={\the\glslongtok},%
9426   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
9427     \protect\glxtrfullsep{\the\glslabeltok}%
9428     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9429   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
9430     \protect\glxtrfullsep{\the\glslabeltok}%
9431     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9432   text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
9433   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
9434 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9435   \glshasattribute{\the\glslabeltok}{regular}%
9436   {%
9437     \glssetattribute{\the\glslabeltok}{regular}{false}%
9438   }%
9439   {}%
9440 }%
9441 }%
9442 {%
```

```
9443 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9444 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9445 \renewcommand*\abbrvpluralsuffix{\glxtrsmsuffix}%
9446 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9447 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
9448 \renewcommand*{\glxtrfullformat}[2]{%
9449   \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9450   \ifglxtrininsertinside\else##2\fi
9451   \glxtrfullsep{##1}%
9452   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9453 }%
9454 \renewcommand*{\glxtrfullplformat}[2]{%
9455   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9456   \ifglxtrininsertinside\else##2\fi
9457   \glxtrfullsep{##1}%
9458   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9459 }%
9460 \renewcommand*{\Glsxtrfullformat}[2]{%
9461   \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9462   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
```

```

9463 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
9464 }%
9465 \renewcommand*{\Glsxtrfullplformat}[2]{%
9466 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9467 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9468 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
9469 }%
9470 }

```

rt-sm-long-desc As before but user provides description

```

9471 \newabbreviationstyle{short-sm-long-desc}%
9472 {%

```

Set accessibility attributes if enabled.

```

9473 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

9474 \renewcommand*{\CustomAbbreviationFields}{%
9475 name={\glxtrshortlongdescname},
9476 sort={\glxtrshortlongdescsort},
9477 first={\protect\glsfirstabbrvsmfont{\the\glssshorttok}%
9478 \protect\glxtrfullsep{\the\glslabeltok}%
9479 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
9480 firstplural={\protect\glsfirstabbrvsmfont{\the\glssshortpltok}%
9481 \protect\glxtrfullsep{\the\glslabeltok}%
9482 \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
9483 text={\protect\glsabbrvsmfont{\the\glssshorttok}},%
9484 plural={\protect\glsabbrvsmfont{\the\glssshortpltok}}}%
9485 }%

```

Unset the regular attribute if it has been set.

```

9486 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9487 \glshasattribute{\the\glslabeltok}{regular}%
9488 {%
9489 \glissetattribute{\the\glslabeltok}{regular}{false}%
9490 }%
9491 {}%
9492 }%
9493 }%
9494 {%

```

As short-sm-long style:

```

9495 \GlsXtrUseAbbrStyleFmts{short-sm-long}%
9496 }

```

short-sm

```

9497 \newabbreviationstyle{short-sm}%
9498 {%

```

Set accessibility attributes if enabled.

```

9499 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```


Setup the default fields.

```

9500 \renewcommand*{\CustomAbbreviationFields}{%
9501   name={\glxtrshortnolongname},
9502   sort={\the\glsshorttok},
9503   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
9504   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
9505   text={\protect\glsabbrvsmfont{\the\glsshorttok}},
9506   plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
9507   description={\the\glslongtok}}%
9508 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9509   \glsetattribute{\the\glslabeltok}{regular}{true}}%
9510 }%
9511 {%
9512 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9513 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9514 \renewcommand*\abbrvpluralsuffix{\glxtrsmsuffix}%
9515 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9516 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

9517 \renewcommand*{\glxtrinlinefullformat}[2]{%
9518   \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}%
9519   \ifglxtrininsertinside##2\fi}%
9520   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9521   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9522 }%
9523 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9524   \protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}%
9525   \ifglxtrininsertinside##2\fi}%
9526   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9527   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9528 }%
9529 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9530   \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}}%
9531   \ifglxtrininsertinside##2\fi}%
9532   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9533   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9534 }%
9535 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9536   \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}%
9537   \ifglxtrininsertinside##2\fi}%
9538   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9539   \glxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9540 }%

```

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.

```

9541 \renewcommand*{\glxtrfullformat}[2]{%

```

```

9542 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9543 \ifglxtrinsertinside\else##2\fi
9544 }%
9545 \renewcommand*{\glxtrfullplformat}[2]{%
9546 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9547 \ifglxtrinsertinside\else##2\fi
9548 }%
9549 \renewcommand*{\Glsxtrfullformat}[2]{%
9550 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9551 \ifglxtrinsertinside\else##2\fi
9552 }%
9553 \renewcommand*{\Glsxtrfullplformat}[2]{%
9554 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9555 \ifglxtrinsertinside\else##2\fi
9556 }%
9557 }

```

short-sm-nolong

```

9558 \letabbreviationstyle{short-sm-nolong}{short-sm}

```

short-sm-desc

```

9559 \newabbreviationstyle{short-sm-desc}%
9560 {%

```

Set accessibility attributes if enabled.

```

9561 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

9562 \renewcommand*{\CustomAbbreviationFields}{%
9563 name={\glxtrshortdescname},
9564 sort={\the\glsshorttok},
9565 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
9566 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
9567 text={\protect\glsabbrvsmfont{\the\glsshorttok}},
9568 plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
9569 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9570 \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
9571 }%
9572 {%

```

```

9573 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9574 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9575 \renewcommand*{\abbrvpluralsuffix}{\glxtrsmsuffix}%
9576 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
9577 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

9578 \renewcommand*{\glxtrinlinefullformat}[2]{%
9579 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9580 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9581 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9582 }%
9583 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
9584 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9585 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9586 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9587 }%
9588 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9589 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9590 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9591 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
9592 }%
9593 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9594 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9595 \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9596 \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
9597 }%

```

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.

```

9598 \renewcommand*{\glsxtrfullformat}[2]{%
9599 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9600 \ifglsxtrininsertinside\else##2\fi
9601 }%
9602 \renewcommand*{\glsxtrfullplformat}[2]{%
9603 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9604 \ifglsxtrininsertinside\else##2\fi
9605 }%
9606 \renewcommand*{\Glsxtrfullformat}[2]{%
9607 \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
9608 \ifglsxtrininsertinside\else##2\fi
9609 }%
9610 \renewcommand*{\Glsxtrfullplformat}[2]{%
9611 \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
9612 \ifglsxtrininsertinside\else##2\fi
9613 }%
9614 }

```

-sm-nolong-desc

```

9615 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

```

nolong-short-sm

```

9616 \newabbreviationstyle{nolong-short-sm}%
9617 {%
9618 \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
9619 }%
9620 {%
9621 \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

9622 \renewcommand*{\glxtrinlinefullformat}[2]{%
9623   \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
9624     \ifglxtrinsertinside##2\fi}%
9625   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9626   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9627 }%
9628 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9629   \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
9630     \ifglxtrinsertinside##2\fi}%
9631   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9632   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9633 }%
9634 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9635   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
9636     \ifglxtrinsertinside##2\fi}%
9637   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9638   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9639 }%
9640 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9641   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
9642     \ifglxtrinsertinside##2\fi}%
9643   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9644   \glxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9645 }%
9646 }

```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```

9647 \newabbreviationstyle{long-noshort-sm}%
9648 {%

```

Set accessibility attributes if enabled.

```
9649 \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```

9650 \renewcommand*{\CustomAbbreviationFields}{%
9651   name={\glxtrlongnoshortname},
9652   sort={\the\glsshorttok},
9653   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
9654   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
9655   text={\protect\glslongdefaultfont{\the\glslongtok}},
9656   plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
9657   description={\the\glslongtok}%
9658 }%
9659 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9660   \glssetattribute{\the\glslabeltok}{regular}{true}}%
9661 }%
9662 {%

```

```
9663 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
```

```

9664 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9665 \renewcommand*\glsabbrvpluralsuffix{\glsxtrsmsuffix}%
9666 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
9667 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9668 \renewcommand*\glsxtrsubsequentfmt[2]{%
9669   \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9670   \ifglsxtrininsertinside \else##2\fi
9671 }%
9672 \renewcommand*\glsxtrsubsequentplfmt[2]{%
9673   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9674   \ifglsxtrininsertinside \else##2\fi
9675 }%
9676 \renewcommand*\Glsxtrsubsequentfmt[2]{%
9677   \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
9678   \ifglsxtrininsertinside \else##2\fi
9679 }%
9680 \renewcommand*\Glsxtrsubsequentplfmt[2]{%
9681   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
9682   \ifglsxtrininsertinside \else##2\fi
9683 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9684 \renewcommand*\glsxtrinlinefullformat[2]{%
9685   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9686   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9687   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9688 }%
9689 \renewcommand*\glsxtrinlinefullplformat[2]{%
9690   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9691   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9692   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9693 }%
9694 \renewcommand*\Glsxtrinlinefullformat[2]{%
9695   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9696   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9697   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9698 }%
9699 \renewcommand*\Glsxtrinlinefullplformat[2]{%
9700   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
9701   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
9702   \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9703 }%

```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```

9704 \renewcommand*\glsxtrfullformat[2]{%
9705   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
9706   \ifglsxtrininsertinside\else##2\fi

```

```

9707 }%
9708 \renewcommand*{\glxtrfullplformat}[2]{%
9709   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9710   \ifglxtrinsertinside\else##2\fi
9711 }%
9712 \renewcommand*{\Glsxtrfullformat}[2]{%
9713   \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside##2\fi}%
9714   \ifglxtrinsertinside\else##2\fi
9715 }%
9716 \renewcommand*{\Glsxtrfullplformat}[2]{%
9717   \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
9718   \ifglxtrinsertinside\else##2\fi
9719 }%
9720 }

```

long-sm Backward compatibility:

```

9721 \@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 \glssshort.

```

9722 \newabbreviationstyle{long-noshort-sm-desc}%
9723 {%
9724   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
9725 }%
9726 {%
9727   \renewcommand*\glssabbrvfont[1]{\glssabbrvsmfont{##1}}%
9728   \renewcommand*\glssfirstabbrvfont[1]{\glssfirstabbrvsmfont{##1}}%
9729   \renewcommand*\glssabbrvpluralsuffix{\glssxtrsmsuffix}%
9730   \renewcommand*\glssfirstlongfont[1]{\glssfirstlongdefaultfont{##1}}%
9731   \renewcommand*\glsslongfont[1]{\glsslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

9732 \renewcommand*{\glxtrsubsequentfnt}[2]{%
9733   \glsslongdefaultfont{\glssaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9734   \ifglxtrinsertinside \else##2\fi
9735 }%
9736 \renewcommand*{\glxtrsubsequentplfnt}[2]{%
9737   \glsslongdefaultfont{\glssaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9738   \ifglxtrinsertinside \else##2\fi
9739 }%
9740 \renewcommand*{\Glsxtrsubsequentfnt}[2]{%
9741   \glsslongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
9742   \ifglxtrinsertinside \else##2\fi
9743 }%
9744 \renewcommand*{\Glsxtrsubsequentplfnt}[2]{%
9745   \glsslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
9746   \ifglxtrinsertinside \else##2\fi
9747 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

9748 \renewcommand*{\glxtrinlinefullformat}[2]{%
9749   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9750   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9751   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
9752 }%
9753 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9754   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9755   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9756   \glxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
9757 }%
9758 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9759   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9760   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9761   \glxtrparen{\protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}}}%
9762 }%
9763 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9764   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9765   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9766   \glxtrparen{\protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}}}%
9767 }%

```

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.

```

9768 \renewcommand*{\glxtrfullformat}[2]{%
9769   \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9770   \ifglxtrininsertinside\else##2\fi
9771 }%
9772 \renewcommand*{\glxtrfullplformat}[2]{%
9773   \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9774   \ifglxtrininsertinside\else##2\fi
9775 }%
9776 \renewcommand*{\Glsxtrfullformat}[2]{%
9777   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
9778   \ifglxtrininsertinside\else##2\fi
9779 }%
9780 \renewcommand*{\Glsxtrfullplformat}[2]{%
9781   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
9782   \ifglxtrininsertinside\else##2\fi
9783 }%
9784 }

```

long-desc-sm Backward compatibility:

```

9785 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

9786 \newabbreviationstyle{short-sm-footnote}%
9787 {%

```

Set accessibility attributes if enabled.

```
9788 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
9789 \renewcommand*{\CustomAbbreviationFields}{%
9790   name={\glxtrfootnotename},
9791   sort={\the\glsshorttok},
9792   description={\the\gslongtok},%
9793   first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
9794     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9795     {\protect\glsfirstlongfootnotefont{\the\gslongtok}}},%
9796   firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
9797     \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9798     {\protect\glsfirstlongfootnotefont{\the\gslongpltok}}},%
9799   text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
9800   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.

```
9801 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9802   \glsssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
9803   \glshasattribute{\the\glslabeltok}{regular}%
9804   {%
9805     \glsssetattribute{\the\glslabeltok}{regular}{false}%
9806   }%
9807   {}%
9808 }%
9809 }%
9810 {%

9811 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvsmfont{##1}}%
9812 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvsmfont{##1}}%
9813 \renewcommand*{\abbrvpluralsuffix}{\glxtrsmsuffix}%
9814 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9815 \renewcommand*{\gslongfont}[1]{\gslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
9816 \renewcommand*{\glxtrfullformat}[2]{%
9817   \glsfirstabbrvsmfont{\glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
9818   \ifglxtrininsertinside\else##2\fi
9819   \protect\glxtrabbrvfootnote{##1}%
9820     {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9821 }%
9822 \renewcommand*{\glxtrfullplformat}[2]{%
9823   \glsfirstabbrvsmfont{\glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
9824   \ifglxtrininsertinside\else##2\fi
9825   \protect\glxtrabbrvfootnote{##1}%
9826     {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9827 }%
9828 \renewcommand*{\Glsxtrfullformat}[2]{%
```



```

9829 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9830 \ifglxtrinsertinside\else##2\fi
9831 \protect\glxtrabbrvfootnote{##1}%
9832 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9833 }%
9834 \renewcommand*{\Glsxtrfullplformat}[2]{%
9835 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9836 \ifglxtrinsertinside\else##2\fi
9837 \protect\glxtrabbrvfootnote{##1}%
9838 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9839 }%

```

The first use full form and the inline full form use the short (long) style.

```

9840 \renewcommand*{\glxtrinlinefullformat}[2]{%
9841 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9842 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9843 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9844 }%
9845 \renewcommand*{\glxtrinlinefullplformat}[2]{%
9846 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9847 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9848 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9849 }%
9850 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9851 \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
9852 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9853 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
9854 }%
9855 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9856 \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
9857 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
9858 \glxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
9859 }%
9860 }

```

footnote-sm Backward compatibility:

```
9861 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}
```

m-footnote-desc Like short-footnote but with user supplied description.

```

9862 \newabbreviationstyle{short-sm-footnote-desc}%
9863 {%

```

Set accessibility attributes if enabled.

```
9864 \glxtrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```

9865 \renewcommand*{\CustomAbbreviationFields}{%
9866 name={\glxtrfootnotedesname},
9867 sort={\glxtrfootnotedesort},
9868 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}}%

```

```

9869 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9870 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
9871 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
9872 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
9873 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
9874 text={\protect\glsabbrvsmfont{\the\glsshorttok}}},%
9875 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.

```

9876 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9877 \glsssetAttribute{\the\glslabeltok}{nohyperfirst}{true}%
9878 \glshasattribute{\the\glslabeltok}{regular}%
9879 {%
9880 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
9881 }%
9882 {}%
9883 }%
9884}%
9885{%
9886 \GlsXtrUseAbbrStyleFmts{short-sm-footnote}%
9887}

```

sm-postfootnote

```

9888 \newabbreviationstyle{short-sm-postfootnote}%
9889 {%

```

Set accessibility attributes if enabled.

```

9890 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

9891 \renewcommand*{\CustomAbbreviationFields}{%
9892 name={\glxtrfootnotename},
9893 sort={\the\glsshorttok},
9894 description={\the\glslongtok},%
9895 first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
9896 firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
9897 text={\protect\glsabbrvsmfont{\the\glsshorttok}}},%
9898 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.

```

9899 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9900 \csdef{glxtrpostlink\glscategorylabel}{%
9901 \glxtrifwasfirstuse
9902 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9903 \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}}%
9904 {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%

```

```

9905     }%
9906     {}%
9907     }%
9908     \glshasattribute{\the\glslabeltok}{regular}%
9909     {%
9910         \glissetattribute{\the\glslabeltok}{regular}{false}%
9911     }%
9912     {}%
9913 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9914 \renewcommand*\glxtrsetupfulldefs{%
9915     \let\glxtrifwasfirstuse\@secondoftwo
9916 }%
9917 }%
9918 {%
9919 \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
9920 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
9921 \renewcommand*\{abbrvpluralsuffix}{\glxtrsmsuffix}%
9922 \renewcommand*\{glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
9923 \renewcommand*\{glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

9924 \renewcommand*\glxtrfullformat[2]{%
9925     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9926     \ifglxtrininsertinside\else##2\fi
9927 }%
9928 \renewcommand*\glxtrfullplformat[2]{%
9929     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9930     \ifglxtrininsertinside\else##2\fi
9931 }%
9932 \renewcommand*\Glsxtrfullformat[2]{%
9933     \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9934     \ifglxtrininsertinside\else##2\fi
9935 }%
9936 \renewcommand*\Glsxtrfullplformat[2]{%
9937     \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9938     \ifglxtrininsertinside\else##2\fi
9939 }%

```

The first use full form and the inline full form use the short (long) style.

```

9940 \renewcommand*\glxtrininlinefullformat[2]{%
9941     \glsfirstabbrvsmfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
9942     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
9943     \glxtrparen{\glsfirstlongfootnotefont{\glssaccesslong{##1}}}%
9944 }%
9945 \renewcommand*\glxtrininlinefullplformat[2]{%
9946     \glsfirstabbrvsmfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
9947     \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

9948 \glstrparen{\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
9949 }%
9950 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
9951 \glfirstabbrvsmfont{\Glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
9952 \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9953 \glstrparen{\glfirstlongfootnotefont{\glaccesslong{##1}}}%
9954 }%
9955 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
9956 \glfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglstrinsertinside##2\fi}%
9957 \ifglstrinsertinside\else##2\fi\glstrfullsep{##1}%
9958 \glstrparen{\glfirstlongfootnotefont{\glaccesslongpl{##1}}}%
9959 }%
9960 }

```

postfootnote-sm Backward compatibility:

```

9961 \@glstr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

stfootnote-desc Like short-sm-postfootnote but with user supplied description.

```

9962 \newabbreviationstyle{short-sm-postfootnote-desc}%
9963 {%

```

Set accessibility attributes if enabled.

```

9964 \glstrAccSuppAbbrSetNameLongAttrs\glscategorylabel

```

Setup the default fields.

```

9965 \renewcommand*{\CustomAbbreviationFields}{%
9966 name={\glstrfootnotedesname},
9967 sort={\glstrfootnotedesort},
9968 first={\protect\glfirstabbrvsmfont{\the\glsshorttok}},%
9969 firstplural={\protect\glfirstabbrvsmfont{\the\glsshortpltok}},%
9970 text={\protect\glabbrvsmfont{\the\glsshorttok}},%
9971 plural={\protect\glabbrvsmfont{\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.

```

9972 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
9973 \csdef{glstrpostlink\glscategorylabel}{%
9974 \glstrifwasfirstuse
9975 {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

9976 \glstrdopostpunc{\protect\glstrabbrvfootnote{\glslabel}%
9977 {\glfirstlongfootnotefont{\glentrylong{\glslabel}}}}%
9978 }%
9979 {}%
9980 }%
9981 \glshasattribute{\the\glslabeltok}{regular}%
9982 {%
9983 \glissetattribute{\the\glslabeltok}{regular}{false}%

```

```

9984 }%
9985 {}%
9986 }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

9987 \renewcommand*{\glxtrsetupfulldefs}{%
9988   \let\glxtrifwasfirstuse\@secondoftwo
9989 }%
9990}%
9991{%
9992 \GlsXtrUseAbbrStyleFmts{short-sm-postfootnote}%
9993}

```

1.7.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```

9994 \newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%

```

`\glsfirstabbrvemfont`

```

9995 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%

```

The default short form suffix:

`\glxxtremsuffix`

```

9996 \newcommand*{\glxxtremsuffix}{\glxtrabbrvpluralsuffix}

```

`\glsfirstlongemfont`

Only used by the “long-em” styles.

```

9997 \newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{#1}}%

```

`\glslongemfont`

Only used by the “long-em” styles.

```

9998 \newcommand*{\glslongemfont}[1]{\emph{#1}}%

```

`\long-short-em`

The long form is just set in the default long font.

```

9999 \newabbreviationstyle{long-short-em}%
10000{}%

```

Set accessibility attributes if enabled.

```

10001 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

10002 \renewcommand*{\CustomAbbreviationFields}{%
10003   name={\glxtrlongshortname},
10004   sort={\the\glsshorttok},
10005   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}}%
10006   \protect\glxtrfullsep{\the\glslabeltok}}%
10007   \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
10008   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}}%

```

```

10009 \protect\glxtrfullsep{\the\glslabeltok}%
10010 \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
10011 text={\protect\glssabbbrvemfont{\the\glsshortttok}}},%
10012 plural={\protect\glssabbbrvemfont{\the\glsshortpltok}}},%
10013 description={\the\glslongtok}}%
10014 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10015 \glshasattribute{\the\glslabeltok}{regular}}%
10016 {%
10017 \glissetattribute{\the\glslabeltok}{regular}{false}}%
10018 }%
10019 {}%
10020 }%
10021 }%
10022 {%

10023 \renewcommand*\glssabbbrvfont[1]{\glssabbbrvemfont{##1}}%
10024 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
10025 \renewcommand*{\abbrvpluralsuffix}{\glstxtremsuffix}%

```

Use the default long fonts.

```

10026 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
10027 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10028 \renewcommand*{\glxtrfullformat}[2]{%
10029 \glsfirstlongdefaultfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10030 \ifglxtrininsertinside\else##2\fi
10031 \glxtrfullsep{##1}}%
10032 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
10033 }%
10034 \renewcommand*{\glxtrfullplformat}[2]{%
10035 \glsfirstlongdefaultfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10036 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
10037 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
10038 }%
10039 \renewcommand*{\Glsxtrfullformat}[2]{%
10040 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10041 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
10042 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
10043 }%
10044 \renewcommand*{\Glsxtrfullplformat}[2]{%
10045 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10046 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}}%
10047 \glxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
10048 }%
10049 }

```

g-short-em-desc

```

10050 \newabbreviationstyle{long-short-em-desc}%
10051 {%

```

Set accessibility attributes if enabled.

```
10052 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```
10053 \renewcommand*{\CustomAbbreviationFields}{%
10054   name={\glxtrlongshortdescname},
10055   sort={\glxtrlongshortdescsort},%
10056   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
10057     \protect\glxtrfullsep{\the\glslabeltok}%
10058     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
10059   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
10060     \protect\glxtrfullsep{\the\glslabeltok}%
10061     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
10062   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10063   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
10064 }%
```

Unset the regular attribute if it has been set.

```
10065 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10066   \glshasattribute{\the\glslabeltok}{regular}%
10067   {%
10068     \glissetattribute{\the\glslabeltok}{regular}{false}%
10069   }%
10070   {}%
10071 }%
10072 }%
10073 {%
```

As long-short-em style:

```
10074 \GlsXtrUseAbbrStyleFmts{long-short-em}%
10075 }
```

ong-em-short-em

```
10076 \newabbreviationstyle{long-em-short-em}%
10077 {%
```

Set accessibility attributes if enabled.

```
10078 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields. \glslongemfont is used in the description since \glstdesc doesn't set the style.

```
10079 \renewcommand*{\CustomAbbreviationFields}{%
10080   name={\glxtrlongshortname},
10081   sort={\the\glsshorttok},
10082   first={\protect\glsfirstlongemfont{\the\glslongtok}%
10083     \protect\glxtrfullsep{\the\glslabeltok}%
10084     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
10085   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
10086     \protect\glxtrfullsep{\the\glslabeltok}%
10087     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
```

```

10088   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10089   plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
10090   description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

10091   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10092     \glsattribute{\the\glslabeltok}{regular}%
10093     {%
10094       \glssetattribute{\the\glslabeltok}{regular}{false}%
10095     }%
10096   }%
10097 }%
10098 }%
10099 {%

10100   \renewcommand*{\abbrvpluralsuffix}{\glstremsuffix}%
10101   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10102   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10103   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
10104   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10105   \renewcommand*{\glsxtrfullformat}[2]{%
10106     \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10107     \ifglsxtrininsertinside\else##2\fi
10108     \glsxtrfullsep{##1}%
10109     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10110   }%
10111   \renewcommand*{\glsxtrfullplformat}[2]{%
10112     \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10113     \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10114     \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10115   }%
10116   \renewcommand*{\Glsxtrfullformat}[2]{%
10117     \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10118     \ifglsxtrininsertinside\else##2\fi\Glsxtrfullsep{##1}%
10119     \Glsxtrparen{\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
10120   }%
10121   \renewcommand*{\Glsxtrfullplformat}[2]{%
10122     \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10123     \ifglsxtrininsertinside\else##2\fi\Glsxtrfullsep{##1}%
10124     \Glsxtrparen{\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
10125   }%
10126 }

```

m-short-em-desc

```

10127 \newabbreviationstyle{long-em-short-em-desc}%
10128 {%

```

Set accessibility attributes if enabled.

```

10129   \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```


Setup the default fields.

```

10130 \renewcommand*{\CustomAbbreviationFields}{%
10131   name={\glxtrlongshortdescname},
10132   sort={\glxtrlongshortdescsort},%
10133   first={\protect\glsfirstlongemfont{\the\glslongtok}%
10134     \protect\glxtrfullsep{\the\glslabeltok}%
10135     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
10136   firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
10137     \protect\glxtrfullsep{\the\glslabeltok}%
10138     \glxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
10139   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10140   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
10141 }%

```

Unset the regular attribute if it has been set.

```

10142 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10143   \glshasattribute{\the\glslabeltok}{regular}%
10144   {%
10145     \glissetattribute{\the\glslabeltok}{regular}{false}%
10146   }%
10147   {}%
10148 }%
10149 }%
10150 {%
10151   \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
10152 }

```

short-em-long Now the short (long) version

```

10153 \newabbreviationstyle{short-em-long}%
10154 {%

```

Set accessibility attributes if enabled.

```

10155 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

10156 \renewcommand*{\CustomAbbreviationFields}{%
10157   name={\glxtrshortlongname},
10158   sort={\the\glsshorttok},
10159   description={\the\glslongtok},%
10160   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10161     \protect\glxtrfullsep{\the\glslabeltok}%
10162     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
10163   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10164     \protect\glxtrfullsep{\the\glslabeltok}%
10165     \glxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
10166   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10167   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

10168 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10169   \glshasattribute{\the\glslabeltok}{regular}%

```

```

10170   {%
10171     \glsetattribute{\the\glslabelltok}{regular}{false}%
10172   }%
10173   {}%
10174 }%
10175 }%
10176 {%

```

Mostly as short-long style:

```

10177 \renewcommand*{\abbrvpluralsuffix}{\glstremsuffix}%
10178 \renewcommand*{\glabbrvfont}[1]{\glabbrvemfont{##1}}%
10179 \renewcommand*{\glfirstabbrvfont}[1]{\glfirstabbrvemfont{##1}}%
10180 \renewcommand*{\glfirstlongfont}[1]{\glfirstlongdefaultfont{##1}}%
10181 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10182 \renewcommand*{\glxtrfullformat}[2]{%
10183   \glfirstabbrvemfont{\glaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10184   \ifglxtrininsertinside\else##2\fi
10185   \glxtrfullsep{##1}%
10186   \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
10187 }%
10188 \renewcommand*{\glxtrfullplformat}[2]{%
10189   \glfirstabbrvemfont{\glaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10190   \ifglxtrininsertinside\else##2\fi
10191   \glxtrfullsep{##1}%
10192   \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
10193 }%
10194 \renewcommand*{\Glsxtrfullformat}[2]{%
10195   \glfirstabbrvemfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10196   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10197   \glxtrparen{\glfirstlongdefaultfont{\glaccesslong{##1}}}%
10198 }%
10199 \renewcommand*{\Glsxtrfullplformat}[2]{%
10200   \glfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10201   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10202   \glxtrparen{\glfirstlongdefaultfont{\glaccesslongpl{##1}}}%
10203 }%
10204 }

```

rt-em-long-desc As before but user provides description

```

10205 \newabbreviationstyle{short-em-long-desc}%
10206 {%

```

Set accessibility attributes if enabled.

```

10207 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

10208 \renewcommand*{\CustomAbbreviationFields}{%
10209   name={\glxtrshortlongdescname},
10210   sort={\glxtrshortlongdescsort},

```

```

10211 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10212 \protect\glstrfullsep{\the\glslabeltok}%
10213 \glstrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
10214 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10215 \protect\glstrfullsep{\the\glslabeltok}%
10216 \glstrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
10217 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10218 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
10219 }%

```

Unset the regular attribute if it has been set.

```

10220 \renewcommand*\GlsXtrPostNewAbbreviation{%
10221 \glshasattribute{\the\glslabeltok}{regular}%
10222 {%
10223 \glissetattribute{\the\glslabeltok}{regular}{false}%
10224 }%
10225 {}%
10226 }%
10227 }%
10228 {%
10229 \GlsXtrUseAbbrStyleFmts{short-em-long}%
10230 }

```

hort-em-long-em

```

10231 \newabbreviationstyle{short-em-long-em}%
10232 {%

```

Set accessibility attributes if enabled.

```

10233 \glstrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields. \glslongemfont is used in the description since \glsdesc doesn't set the style.

```

10234 \renewcommand*\CustomAbbreviationFields{%
10235 name={\glstrshortlongname},
10236 sort={\the\glsshorttok},
10237 description={\protect\glslongemfont{\the\glslongtok}},%
10238 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10239 \protect\glstrfullsep{\the\glslabeltok}%
10240 \glstrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
10241 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10242 \protect\glstrfullsep{\the\glslabeltok}%
10243 \glstrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
10244 text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10245 plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%

```

Unset the regular attribute if it has been set.

```

10246 \renewcommand*\GlsXtrPostNewAbbreviation{%
10247 \glshasattribute{\the\glslabeltok}{regular}%
10248 {%
10249 \glissetattribute{\the\glslabeltok}{regular}{false}%

```

```

10250 }%
10251 {}%
10252 }%
10253 }%
10254 {%

10255 \renewcommand*{\abbrvpluralsuffix}{\glxtremsuffix}%
10256 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10257 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10258 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
10259 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

10260 \renewcommand*{\glsxtrfullformat}[2]{%
10261   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10262   \ifglsxtrininsertinside\else##2\fi
10263   \glsxtrfullsep{##1}%
10264   \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
10265 }%
10266 \renewcommand*{\glsxtrfullplformat}[2]{%
10267   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10268   \ifglsxtrininsertinside\else##2\fi
10269   \glsxtrfullsep{##1}%
10270   \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
10271 }%
10272 \renewcommand*{\Glsxtrfullformat}[2]{%
10273   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10274   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10275   \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
10276 }%
10277 \renewcommand*{\Glsxtrfullplformat}[2]{%
10278   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10279   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10280   \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
10281 }%
10282 }

```

em-long-em-desc

```

10283 \newabbreviationstyle{short-em-long-em-desc}%
10284 {%

```

Set accessibility attributes if enabled.

```

10285 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

10286 \renewcommand*{\CustomAbbreviationFields}{%
10287   name={\glsxtrshortlongdescname},%
10288   sort={\glsxtrshortlongdescsort},%
10289   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}}%
10290   \protect\glsxtrfullsep{\the\glslabeltok}}%
10291   \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%

```

```

10292   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10293   \protect\glsxtrfullsep{\the\glslabeltok}%
10294   \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
10295   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10296   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
10297 }%

```

Unset the regular attribute if it has been set.

```

10298 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10299   \glsasattribute{\the\glslabeltok}{regular}%
10300   {%
10301     \glssetattribute{\the\glslabeltok}{regular}{false}%
10302   }%
10303   {}%
10304 }%
10305}%
10306{%
10307   \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
10308 }

```

short-em

```

10309 \newabbreviationstyle{short-em}%
10310 {%

```

Set accessibility attributes if enabled.

```

10311   \glsxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

10312 \renewcommand*{\CustomAbbreviationFields}{%
10313   name={\glsxtrshortnolongname},
10314   sort={\the\glsshorttok},
10315   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
10316   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
10317   text={\protect\glsabbrvemfont{\the\glsshorttok}},
10318   plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
10319   description={\the\glslongtok}}%
10320 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10321   \glssetattribute{\the\glslabeltok}{regular}{true}}%
10322}%
10323{%

10324 \renewcommand*{\abbrvpluralsuffix}{\glsxtrsuffix}%
10325 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10326 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10327 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
10328 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

10329 \renewcommand*{\glxtrinlinefullformat}[2]{%
10330   \protect\glsfirstabbrvemfont{\glsaccessshort{##1}}%
10331   \ifglxtrininsertinside##2\fi}%

```

```

10332 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10333 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
10334 }%
10335 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10336 \protect\glsfirstabbrvemfont{\glssaccessshortpl{##1}%
10337 \ifglxtrinsertinside##2\fi}%
10338 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10339 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
10340 }%

10341 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10342 \protect\glsfirstabbrvemfont{\Glsaccessshort{##1}%
10343 \ifglxtrinsertinside##2\fi}%
10344 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10345 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
10346 }%
10347 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10348 \protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}%
10349 \ifglxtrinsertinside##2\fi}%
10350 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10351 \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
10352 }%

```

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.

```

10353 \renewcommand*{\glxtrfullformat}[2]{%
10354 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10355 \ifglxtrinsertinside\else##2\fi
10356 }%
10357 \renewcommand*{\glxtrfullplformat}[2]{%
10358 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10359 \ifglxtrinsertinside\else##2\fi
10360 }%
10361 \renewcommand*{\Glsxtrfullformat}[2]{%
10362 \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
10363 \ifglxtrinsertinside\else##2\fi
10364 }%
10365 \renewcommand*{\Glsxtrfullplformat}[2]{%
10366 \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
10367 \ifglxtrinsertinside\else##2\fi
10368 }%
10369 }

```

short-em-nolong

```
10370 \letabbreviationstyle{short-em-nolong}{short-em}
```

short-em-desc

```

10371 \newabbreviationstyle{short-em-desc}%
10372 {}

```

Set accessibility attributes if enabled. The default name includes the long form but \glxtrshortdesname could be modified to omit the long form, so include the nameshortaccess attribute.

```
10373 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
```

Setup the default fields.

```
10374 \renewcommand*{\CustomAbbreviationFields}{%
10375   name={\glxtrshortdesname},
10376   sort={\the\glsshorttok},
10377   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
10378   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
10379   text={\protect\glsabbrvemfont{\the\glsshorttok}},
10380   plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
10381 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10382   \glsssetattribute{\the\glslabeltok}{regular}{true}}%
10383 }%
10384 {%

10385 \renewcommand*{\abbrvpluralsuffix}{\glxtremsuffix}%
10386 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10387 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10388 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
10389 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
10390 \renewcommand*{\glxtrinlinefullformat}[2]{%
10391   \glsfirstabbrvemfont{\glssaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
10392   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10393   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslong{##1}}}%
10394 }%

10395 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10396   \glsfirstabbrvemfont{\glssaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
10397   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10398   \glxtrparen{\glsfirstlongdefaultfont{\glssaccesslongpl{##1}}}%
10399 }%

10400 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10401   \glsfirstabbrvemfont{\Glsaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
10402   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10403   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslong{##1}}}%
10404 }%

10405 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10406   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}}\ifglxtrininsertinside##2\fi}%
10407   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10408   \glxtrparen{\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}}}%
10409 }%
```

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.

```
10410 \renewcommand*{\glxtrfullformat}[2]{%
10411   \glsfirstabbrvemfont{\glssaccessshort{##1}}\ifglxtrininsertinside##2\fi}%
10412   \ifglxtrininsertinside\else##2\fi
```

```

10413 }%
10414 \renewcommand*{\glxtrfullplformat}[2]{%
10415   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10416   \ifglxtrininsertinside\else##2\fi
10417 }%
10418 \renewcommand*{\Glsxtrfullformat}[2]{%
10419   \glsfirstabbrvemfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
10420   \ifglxtrininsertinside\else##2\fi
10421 }%
10422 \renewcommand*{\Glsxtrfullplformat}[2]{%
10423   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
10424   \ifglxtrininsertinside\else##2\fi
10425 }%
10426 }

```

-em-nolong-desc

```

10427 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

nolong-short-em

```

10428 \newabbreviationstyle{nolong-short-em}%
10429 {%
10430   \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
10431 }%
10432 {%
10433   \GlsXtrUseAbbrStyleFmts{short-em-nolong}%

```

The inline full form displays the long form followed by the short form in parentheses.

```

10434 \renewcommand*{\glxtrinlinefullformat}[2]{%
10435   \protect\glsfirstlongdefaultfont{\glssaccesslong{##1}%
10436     \ifglxtrininsertinside##2\fi}%
10437   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10438   \glxtrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
10439 }%
10440 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10441   \protect\glsfirstlongdefaultfont{\glssaccesslongpl{##1}%
10442     \ifglxtrininsertinside##2\fi}%
10443   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10444   \glxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%
10445 }%
10446 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10447   \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
10448     \ifglxtrininsertinside##2\fi}%
10449   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10450   \glxtrparen{\glsfirstabbrvemfont{\glssaccessshort{##1}}}%
10451 }%
10452 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10453   \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
10454     \ifglxtrininsertinside##2\fi}%
10455   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10456   \glxtrparen{\glsfirstabbrvemfont{\glssaccessshortpl{##1}}}%

```



```

10457 }%
10458 }

```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

10459 \newabbreviationstyle{long-noshort-em}%
10460 {%

```

Set accessibility attributes if enabled.

```

10461 \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel

```

Setup the default fields.

```

10462 \renewcommand*{\CustomAbbreviationFields}{%
10463   name={\glxtrlongnoshortname},
10464   sort={\the\glsshorttok},
10465   first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
10466   firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
10467   text={\protect\glslongdefaultfont{\the\glslongtok}},
10468   plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
10469   description={\the\glslongtok}%
10470 }%
10471 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10472   \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
10473 }%
10474 {%
10475   \renewcommand*{\abbrvpluralsuffix}{\glxtremsuffix}%
10476   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10477   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10478   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
10479   \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10480 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10481   \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
10482   \ifglxtrininsertinside \else##2\fi
10483 }%
10484 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10485   \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
10486   \ifglxtrininsertinside \else##2\fi
10487 }%
10488 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10489   \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
10490   \ifglxtrininsertinside \else##2\fi
10491 }%
10492 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10493   \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
10494   \ifglxtrininsertinside \else##2\fi
10495 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10496 \renewcommand*{\glsxtrinlinefullformat}[2]{%

```

```

10497 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10498 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10499 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10500 }%
10501 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10502 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10503 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10504 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10505 }%
10506 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10507 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10508 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10509 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10510 }%
10511 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10512 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10513 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10514 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10515 }%

```

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.

```

10516 \renewcommand*{\glxtrfullformat}[2]{%
10517 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10518 \ifglxtrinsertinside\else##2\fi
10519 }%
10520 \renewcommand*{\glxtrfullplformat}[2]{%
10521 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10522 \ifglxtrinsertinside\else##2\fi
10523 }%
10524 \renewcommand*{\Glsxtrfullformat}[2]{%
10525 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10526 \ifglxtrinsertinside\else##2\fi
10527 }%
10528 \renewcommand*{\Glsxtrfullplformat}[2]{%
10529 \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10530 \ifglxtrinsertinside\else##2\fi
10531 }%
10532 }

```

long-em Backward compatibility:

```
10533 \@glxtr@deprecated@abbrstyle{long-em}{long-noshort-em}
```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```

10534 \newabbreviationstyle{long-em-noshort-em}%
10535 {}%

```

Set accessibility attributes if enabled.

```
10536 \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel
```

Setup the default fields.

```

10537 \renewcommand*{\CustomAbbreviationFields}{%
10538   name={\glxstrlongnoshortname},
10539   sort={\the\glsshorttok},
10540   first={\protect\glxfirstlongemfont{\the\glslongtok}},
10541   firstplural={\protect\glxfirstlongemfont{\the\glslongpltok}},
10542   text={\protect\glslongemfont{\the\glslongtok}},
10543   plural={\protect\glslongemfont{\the\glslongpltok}},%
10544   description={\protect\glslongemfont{\the\glslongtok}}%
10545 }%
10546 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10547   \glxsetattribute{\the\glslabeltok}{regular}{true}}%
10548 }%
10549 {%

10550 \renewcommand*{\abbrvpluralsuffix}{\glxxtremsuffix}%
10551 \renewcommand*{\glxabbrvfont}[1]{\glxabbrvemfont{##1}}%
10552 \renewcommand*{\glxfirstabbrvfont}[1]{\glxfirstabbrvemfont{##1}}%
10553 \renewcommand*{\glxfirstlongfont}[1]{\glxfirstlongemfont{##1}}%
10554 \renewcommand*{\glxlongfont}[1]{\glxlongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10555 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10556   \glxlongemfont{\glxsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
10557   \ifglxtrinsertinside \else##2\fi
10558 }%
10559 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10560   \glxlongemfont{\glxsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
10561   \ifglxtrinsertinside \else##2\fi
10562 }%
10563 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10564   \glxlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside ##2\fi}%
10565   \ifglxtrinsertinside \else##2\fi
10566 }%
10567 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10568   \glxlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside ##2\fi}%
10569   \ifglxtrinsertinside \else##2\fi
10570 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10571 \renewcommand*{\glxtrinlinefullformat}[2]{%
10572   \glxfirstlongemfont{\glxsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10573   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10574   \glxtrparen{\protect\glxfirstabbrvemfont{\glxsaccessshort{##1}}}%
10575 }%
10576 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10577   \glxfirstlongemfont{\glxsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10578   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10579   \glxtrparen{\protect\glxfirstabbrvemfont{\glxsaccessshortpl{##1}}}%
10580 }%

```

```

10581 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10582   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10583   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10584   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshort{##1}}}%
10585 }%
10586 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10587   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10588   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10589   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
10590 }%

```

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.

```

10591 \renewcommand*{\glxtrfullformat}[2]{%
10592   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10593   \ifglxtrinsertinside\else##2\fi
10594 }%
10595 \renewcommand*{\glxtrfullplformat}[2]{%
10596   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10597   \ifglxtrinsertinside\else##2\fi
10598 }%
10599 \renewcommand*{\Glsxtrfullformat}[2]{%
10600   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10601   \ifglxtrinsertinside\else##2\fi
10602 }%
10603 \renewcommand*{\Glsxtrfullplformat}[2]{%
10604   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10605   \ifglxtrinsertinside\else##2\fi
10606 }%
10607 }

```

`noshort-em-noreg` Like `long-em-noshort-em` but doesn't set the regular attribute.

```

10608 \newabbreviationstyle{long-em-noshort-em-noreg}%
10609 {}%

```

Set accessibility attributes if enabled.

```

10610 \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel

```

Setup the default fields.

```

10611 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%

```

Unset the regular attribute if it has been set.

```

10612 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10613   \glshasattribute{\the\glslabeltok}{regular}%
10614   {%
10615     \glissetattribute{\the\glslabeltok}{regular}{false}%
10616   }%
10617   {}%
10618 }%
10619 }%
10620 {}%

```

```

10621 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
10622 }

```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like `\glssshort`.

```

10623 \newabbreviationstyle{long-noshort-em-desc}%
10624 {%
10625 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
10626 }%
10627 {%

10628 \renewcommand*{\abbrvpluralsuffix}{\glstxtremsuffix}%
10629 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10630 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
10631 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
10632 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10633 \renewcommand*{\glxtrsubsequentfmt}[2]{%
10634 \glslongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
10635 \ifglxtrininsertinside \else##2\fi
10636 }%
10637 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
10638 \glslongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
10639 \ifglxtrininsertinside \else##2\fi
10640 }%
10641 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10642 \glslongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside ##2\fi}%
10643 \ifglxtrininsertinside \else##2\fi
10644 }%
10645 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10646 \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside ##2\fi}%
10647 \ifglxtrininsertinside \else##2\fi
10648 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10649 \renewcommand*{\glxtrinlinefullformat}[2]{%
10650 \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10651 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10652 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10653 }%
10654 \renewcommand*{\glxtrinlinefullplformat}[2]{%
10655 \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
10656 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10657 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10658 }%
10659 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10660 \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
10661 \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
10662 \glxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%

```

```

10663 }%
10664 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10665   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10666   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
10667   \glxtrparen{\protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}}}%
10668 }%

```

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.

```

10669 \renewcommand*{\glxtrfullformat}[2]{%
10670   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10671   \ifglxtrinsertinside\else##2\fi
10672 }%
10673 \renewcommand*{\glxtrfullplformat}[2]{%
10674   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10675   \ifglxtrinsertinside\else##2\fi
10676 }%
10677 \renewcommand*{\Glsxtrfullformat}[2]{%
10678   \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10679   \ifglxtrinsertinside\else##2\fi
10680 }%
10681 \renewcommand*{\Glsxtrfullplformat}[2]{%
10682   \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10683   \ifglxtrinsertinside\else##2\fi
10684 }%
10685 }

```

long-desc-em Backward compatibility:

```

10686 \@glxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}

```

noshort-em-desc The short form is explicitly invoked through commands like `\glxtrshort`. The long form is emphasized. No accessibility attributes need to be set.

```

10687 \newabbreviationstyle{long-em-noshort-em-desc}%
10688 {%
10689   \renewcommand*{\CustomAbbreviationFields}{%
10690     name={\glxtrlongnoshortdescname},
10691     sort={\the\glslongtok},
10692     first={\protect\glsfirstlongemfont{\the\glslongtok}},
10693     firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
10694     text={\glslongemfont{\the\glslongtok}},
10695     plural={\glslongemfont{\the\glslongpltok}}}%
10696   }%
10697   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10698     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
10699 }%
10700 {%
10701   \renewcommand*{\abbrvpluralsuffix}{\glxtremsuffix}%
10702   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
10703   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%

```

```

10704 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
10705 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

10706 \renewcommand*{\glsxtrsubsequentfmt}[2]{%
10707   \glslongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
10708   \ifglsxtrininsertinside \else##2\fi
10709 }%
10710 \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
10711   \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
10712   \ifglsxtrininsertinside \else##2\fi
10713 }%
10714 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
10715   \glslongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside ##2\fi}%
10716   \ifglsxtrininsertinside \else##2\fi
10717 }%
10718 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
10719   \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside ##2\fi}%
10720   \ifglsxtrininsertinside \else##2\fi
10721 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

10722 \renewcommand*{\glsxtrinlinefullformat}[2]{%
10723   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10724   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10725   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10726 }%
10727 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
10728   \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10729   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10730   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10731 }%
10732 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10733   \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10734   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10735   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
10736 }%
10737 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10738   \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
10739   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10740   \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
10741 }%

```

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.

```

10742 \renewcommand*{\glsxtrfullformat}[2]{%
10743   \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
10744   \ifglsxtrininsertinside\else##2\fi
10745 }%
10746 \renewcommand*{\glsxtrfullplformat}[2]{%

```

```

10747 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10748 \ifglxtrinsertinside\else##2\fi
10749 }%
10750 \renewcommand*{\Glsxtrfullformat}[2]{%
10751 \glsfirstlongemfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
10752 \ifglxtrinsertinside\else##2\fi
10753 }%
10754 \renewcommand*{\Glsxtrfullplformat}[2]{%
10755 \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
10756 \ifglxtrinsertinside\else##2\fi
10757 }%
10758 }

```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```

10759 \newabbreviationstyle{long-em-noshort-em-desc-noreg}%
10760 {%
10761 \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%
    Unset the regular attribute if it has been set.
10762 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10763 \glshasattribute{\the\glslabeltok}{regular}%
10764 {%
10765 \glssetattribute{\the\glslabeltok}{regular}{false}%
10766 }%
10767 {}%
10768 }%
10769 }%
10770 {%
10771 \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
10772 }

```

ort-em-footnote

```

10773 \newabbreviationstyle{short-em-footnote}%
10774 {%
    Set accessibility attributes if enabled.
10775 \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel
    Setup the default fields.
10776 \renewcommand*{\CustomAbbreviationFields}{%
10777 name={\glxtrfootnotename},
10778 sort={\the\glsshorttok},
10779 description={\the\glslongtok},%
10780 first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
10781 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10782 {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
10783 firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
10784 \protect\glxtrabbrvfootnote{\the\glslabeltok}%
10785 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
10786 text={\protect\glsabbrvemfont{\the\glsshorttok}}},%
10787 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.

```

10788 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10789 \glsssetattribute{\the\glslabelltok}{nohyperfirst}{true}%
10790 \glshasattribute{\the\glslabelltok}{regular}%
10791 {%
10792 \glsssetattribute{\the\glslabelltok}{regular}{false}%
10793 }%
10794 {}%
10795 }%
10796 }%
10797 {%

10798 \renewcommand*{\abbrvpluralsuffix}{\glstxtremsuffix}%
10799 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvemfont{##1}}%
10800 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvemfont{##1}}%
10801 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongfootnotefont{##1}}%
10802 \renewcommand*{\glsslongfont}[1]{\glsslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

10803 \renewcommand*{\glssxtrfullformat}[2]{%
10804 \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
10805 \ifglssxtrininsertinside\else##2\fi
10806 \protect\glssxtrabbrvfootnote{##1}%
10807 {\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
10808 }%
10809 \renewcommand*{\glssxtrfullplformat}[2]{%
10810 \glssfirstabbrvemfont{\glssaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
10811 \ifglssxtrininsertinside\else##2\fi
10812 \protect\glssxtrabbrvfootnote{##1}%
10813 {\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10814 }%
10815 \renewcommand*{\Glsxtrfullformat}[2]{%
10816 \glssfirstabbrvemfont{\Glsaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
10817 \ifglssxtrininsertinside\else##2\fi
10818 \protect\glssxtrabbrvfootnote{##1}%
10819 {\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%
10820 }%
10821 \renewcommand*{\Glsxtrfullplformat}[2]{%
10822 \glssfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglssxtrininsertinside##2\fi}%
10823 \ifglssxtrininsertinside\else##2\fi
10824 \protect\glssxtrabbrvfootnote{##1}%
10825 {\glssfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10826 }%

```

The first use full form and the inline full form use the short (long) style.

```

10827 \renewcommand*{\glssxtrininlinefullformat}[2]{%
10828 \glssfirstabbrvemfont{\glssaccessshort{##1}\ifglssxtrininsertinside##2\fi}%
10829 \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
10830 \glssxtrparen{\glssfirstlongfootnotefont{\glssaccesslong{##1}}}%

```

```

10831 }%
10832 \renewcommand*{\glxstrinlinefullplformat}[2]{%
10833   \glsfirstabbrvemfont{\glssaccessshortpl{##1}\ifglxstrinsertinside##2\fi}%
10834   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
10835   \glxstrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10836 }%
10837 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
10838   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglxstrinsertinside##2\fi}%
10839   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
10840   \glxstrparen{\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
10841 }%
10842 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
10843   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglxstrinsertinside##2\fi}%
10844   \ifglxstrinsertinside\else##2\fi\glxstrfullsep{##1}%
10845   \glxstrparen{\glsfirstlongfootnotefont{\glssaccesslongpl{##1}}}%
10846 }%
10847 }

```

footnote-em Backward compatibility:

```
10848 \@glxstr@deprecated@abbrstyle{footnote-em}{short-em-footnote}
```

m-footnote-desc Like short-em-footnote but with user supplied description.

```

10849 \newabbreviationstyle{short-em-footnote-desc}%
10850 {%

```

Set accessibility attributes if enabled.

```
10851 \glxstrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```

10852 \renewcommand*{\CustomAbbreviationFields}{%
10853   name={\glxstrfootnotedesname},
10854   sort={\glxstrfootnotedesort},
10855   first={\protect\glsfirstabbrvemfont{\the\glssshorttok}%
10856     \protect\glxstrabbrvfootnote{\the\glslabeltok}%
10857     {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
10858   firstplural={\protect\glsfirstabbrvemfont{\the\glssshortpltok}%
10859     \protect\glxstrabbrvfootnote{\the\glslabeltok}%
10860     {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
10861   text={\protect\glsabbrvemfont{\the\glssshorttok}},%
10862   plural={\protect\glsabbrvemfont{\the\glssshortpltok}}}%

```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```

10863 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10864   \glsssetAttribute{\the\glslabeltok}{nohyperfirst}{true}%
10865   \glshasattribute{\the\glslabeltok}{regular}%
10866   {%
10867     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
10868   }%
10869   {}%

```

```

10870 }%
10871 }%
10872 {%
10873   \GlsXtrUseAbbrStyleFmts{short-em-footnote}%
10874 }

```

em-postfootnote

```

10875 \newabbreviationstyle{short-em-postfootnote}%
10876 {%

```

Set accessibility attributes if enabled.

```

10877   \glxtrAccSuppAbbrSetNoLongAttrs\glscategorylabel

```

Setup the default fields.

```

10878   \renewcommand*{\CustomAbbreviationFields}{%
10879     name={\glxtrfootnotename},
10880     sort={\the\glsshorttok},
10881     description={\the\glslongtok},%
10882     first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
10883     firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
10884     text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10885     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.

```

10886   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10887     \csdef{glxtrpostlink\glscategorylabel}{%
10888       \glxtrifwasfirstuse
10889       {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

10890         \glxtrdopostpunc{\protect\glxtrabbrvf footnote{\glslabel}%
10891         {\glsfirstlongfootnotefont{\gl Sentrylong{\glslabel}}}}%
10892       }%
10893     }%
10894   }%
10895   \glshasattribute{\the\glslabeltok}{regular}%
10896   {%
10897     \glissetattribute{\the\glslabeltok}{regular}{false}%
10898   }%
10899   {}%
10900 }%

```

The footnote needs to be suppressed in the inline form, so \glxtrfull must set the first use switch off.

```

10901   \renewcommand*{\glxtrsetupfulldefs}{%
10902     \let\glxtrifwasfirstuse\@secondoftwo
10903   }%
10904 }%
10905 {%

```

```

10906 \renewcommand*\abbrvpluralsuffix{\glxxtremsuffix}%
10907 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
10908 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
10909 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
10910 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

10911 \renewcommand*\glsxtrfullformat[2]{%
10912   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10913   \ifglsxtrininsertinside\else##2\fi
10914 }%
10915 \renewcommand*\glsxtrfullplformat[2]{%
10916   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10917   \ifglsxtrininsertinside\else##2\fi
10918 }%
10919 \renewcommand*\Glsxtrfullformat[2]{%
10920   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10921   \ifglsxtrininsertinside\else##2\fi
10922 }%
10923 \renewcommand*\Glsxtrfullplformat[2]{%
10924   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10925   \ifglsxtrininsertinside\else##2\fi
10926 }%

```

The first use full form and the inline full form use the short (long) style.

```

10927 \renewcommand*\glsxtrininlinefullformat[2]{%
10928   \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10929   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10930   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10931 }%
10932 \renewcommand*\glsxtrininlinefullplformat[2]{%
10933   \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10934   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10935   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10936 }%
10937 \renewcommand*\Glsxtrininlinefullformat[2]{%
10938   \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
10939   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10940   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
10941 }%
10942 \renewcommand*\Glsxtrininlinefullplformat[2]{%
10943   \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
10944   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
10945   \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
10946 }%
10947 }

```

postfootnote-em Backward compatibility:

```

10948 \@glsxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}

```

stfootnote-desc Like short-em-postfootnote but with user supplied description.

```
10949 \newabbreviationstyle{short-em-postfootnote-desc}%
10950 {%
```

Set accessibility attributes if enabled.

```
10951 \glxstrAccSuppAbbrSetNameLongAttrs\glscategorylabel
```

Setup the default fields.

```
10952 \renewcommand*{\CustomAbbreviationFields}{%
10953   name={\glxstrfootnotedesname},
10954   sort={\glxstrfootnotedesort},
10955   first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
10956   firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
10957   text={\protect\glsabbrvemfont{\the\glsshorttok}},%
10958   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.

```
10959 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
10960   \csdef{glxtrpostlink\glscategorylabel}{%
10961     \glxtrifwasfirstuse
10962     {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
10963       \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
10964       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
10965     }%
10966   }%
10967 }%
10968 \glshasattribute{\the\glslabeltok}{regular}%
10969 {%
10970   \glissetattribute{\the\glslabeltok}{regular}{false}%
10971   }%
10972   {}%
10973 }%
```

The footnote needs to be suppressed in the inline form, so \glxtrfull must set the first use switch off.

```
10974 \renewcommand*{\glxtrsetupfulldefs}{%
10975   \let\glxtrifwasfirstuse\@secondoftwo
10976   }%
10977 }%
10978 {%
10979   \GlsXtrUseAbbrStyleFmts{short-em-postfootnote}%
10980 }
```

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.

```
10981 \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.

```
10982 \ifdef\glscurrentfieldvalue
10983 {
10984   \newcommand*{\glxstruserparen}[2]{%
10985     \glxstrfullsep{#2}%
10986     \glxstrparen
10987     {#1\ifglshasfield{\glxstruserfield}{#2}{, \glscurrentfieldvalue}{}}%
10988   }
10989 }
10990 {
10991   \newcommand*{\glxstruserparen}[2]{%
10992     \glxstrfullsep{#2}%
10993     \glxstrparen
10994     {#1\ifglshasfield{\glxstruserfield}{#2}{, \@glo@thisvalue}{}}%
10995   }
10996 }
```

Font used for short form:

`lsabbrvuserfont`

```
10997 \newcommand*{\glabbrvuserfont}[1]{\glabbrvdefaultfont{#1}}
```

Font used for short form on first use:

`stabbrvuserfont`

```
10998 \newcommand*{\glfirstabbrvuserfont}[1]{\glabbrvuserfont{#1}}
```

Font used for long form:

`glslonguserfont`

```
10999 \newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

`rstlonguserfont`

```
11000 \newcommand*{\glfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

`lsxtrusersuffix`

```
11001 \newcommand*{\glxtrusersuffix}{\glxtrabbrvpluralsuffix}
```

Description encapsulator.

`userdescription` The first argument is the description. The second argument is the label.

```
11002 \newcommand*{\gluserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
11003 \newabbreviationstyle{long-short-user}%  
11004 {%
```

Set accessibility attributes if enabled.

```
11005 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
11006 \renewcommand*{\CustomAbbreviationFields}{%  
11007   name={\glxtrlongshortname},  
11008   sort={\the\glsshorttok},  
11009   first={\protect\glsfirstlonguserfont{\the\glslongtok}%  
11010     \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}}%  
11011     {\the\glslabeltok}},%  
11012   firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%  
11013     \protect\glxtruserparen  
11014     {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%  
  
11015   text={\protect\glsabbrvuserfont{\the\glsshorttok}},%  
11016   plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%  
11017   description={\protect\glsuserdescription{\the\glslongtok}%  
11018     {\the\glslabeltok}}}%
```

Unset the regular attribute if it has been set.

```
11019 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
11020   \glshasattribute{\the\glslabeltok}{regular}%  
11021   {%  
11022     \glissetattribute{\the\glslabeltok}{regular}{false}%  
11023   }%  
11024   {}%  
11025 }%  
11026 }%  
11027 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
11028 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%  
11029 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%  
11030 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%  
11031 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%  
11032 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
11033 \renewcommand*{\glxtrfullformat}[2]{%  
11034   \glsfirstlonguserfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%  
11035   \ifglxtrinsertinside\else##2\fi  
11036   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%  
11037 }%  
11038 \renewcommand*{\glxtrfullplformat}[2]{%  
11039   \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%  
11040   \ifglxtrinsertinside\else##2\fi  
11041   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
```

```

11042 }%
11043 \renewcommand*{\Glsxtrfullformat}[2]{%
11044   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11045   \ifglxtrinsertinside\else##2\fi
11046   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}{##1}%
11047 }%
11048 \renewcommand*{\Glsxtrfullplformat}[2]{%
11049   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11050   \ifglxtrinsertinside\else##2\fi
11051   \glxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}{##1}%
11052 }%
11053 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

11054 \newabbreviationstyle{long-postshort-user}%
11055 {%

```

Set accessibility attributes if enabled.

```

11056 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

11057 \renewcommand*{\CustomAbbreviationFields}{%
11058   name={\glxtrlongshortname},
11059   sort={\the\glsshorttok},
11060   first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
11061   firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

11062   text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
11063   plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
11064   description={\protect\glsuserdescription{\the\glslongtok}%
11065     {\the\glslabeltok}}}%
11066 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11067   \csdef{glxtrpostlink\glscategorylabel}{%
11068     \glxtrifwasfirstuse
11069     {%
11070       \glxtruserparen
11071         {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
11072         {\glslabel}%
11073     }%
11074   }%
11075 }%
11076 \glshasattribute{\the\glslabeltok}{regular}%
11077 {%
11078   \glissetattribute{\the\glslabeltok}{regular}{false}%
11079 }%
11080 }%
11081 }%
11082 }%
11083 {%

```

In case the user wants to mix and match font styles, these are redefined here.


```

11084 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
11085 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
11086 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
11087 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
11088 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

11089 \renewcommand*{\glxtrfullformat}[2]{%
11090   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
11091   \ifglxtrininsertinside\else##2\fi
11092 }%
11093 \renewcommand*{\glxtrfullplformat}[2]{%
11094   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
11095   \ifglxtrininsertinside\else##2\fi
11096 }%
11097 \renewcommand*{\Glsxtrfullformat}[2]{%
11098   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
11099   \ifglxtrininsertinside\else##2\fi
11100 }%
11101 \renewcommand*{\Glsxtrfullplformat}[2]{%
11102   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
11103   \ifglxtrininsertinside\else##2\fi
11104 }%

```

In-line format:

```

11105 \renewcommand*{\glxtrinlinefullformat}[2]{%
11106   \glsfirstlonguserfont{\glssaccesslong{##1}\ifglxtrininsertinside##2\fi}%
11107   \ifglxtrininsertinside\else##2\fi
11108   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
11109 }%
11110 \renewcommand*{\glxtrinlinefullplformat}[2]{%
11111   \glsfirstlonguserfont{\glssaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
11112   \ifglxtrininsertinside\else##2\fi
11113   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
11114 }%
11115 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11116   \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglxtrininsertinside##2\fi}%
11117   \ifglxtrininsertinside\else##2\fi
11118   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshort{##1}}}{##1}%
11119 }%
11120 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11121   \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglxtrininsertinside##2\fi}%
11122   \ifglxtrininsertinside\else##2\fi
11123   \glxtruserparen{\glsfirstabbrvuserfont{\glssaccessshortpl{##1}}}{##1}%
11124 }%
11125 }

```

ortuserdesname

```

11126 \newcommand*{\glxtrlongshortuserdesname}{%
11127   \protect\glslonguserfont{\the\glslongtok}%

```

```

11128 \protect\glxtruserparen
11129 {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
11130 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

11131 \newabbreviationstyle{long-postshort-user-desc}%
11132 {%
    Set accessibility attributes if enabled.
11133 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
    Setup the default fields.
11134 \renewcommand*{\CustomAbbreviationFields}{%
11135     name={\glxtrlongshortuserdescname},
11136     sort={\the\glslongtok},
11137     first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
11138     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
11139     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
11140     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
11141 }%
11142 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11143 \csdef{glxtrpostlink\glscategorylabel}{%
11144     \glxtrifwasfirstuse
11145     {%
11146         \glxtruserparen
11147         {\glsfirstabbrvuserfont{\glstryshort{\glslabel}}}%
11148         {\glslabel}%
11149     }%
11150     }%
11151 }%
11152 \glshasattribute{\the\glslabeltok}{regular}%
11153 {%
11154     \glissetattribute{\the\glslabeltok}{regular}{false}%
11155 }%
11156 {}%
11157 }%
11158 }%
11159 {%
11160 \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
11161 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

11162 \newabbreviationstyle{short-postlong-user}%
11163 {%
    Set accessibility attributes if enabled.
11164 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
    Setup the default fields.
11165 \renewcommand*{\CustomAbbreviationFields}{%

```

```

11166   name={\glxtrshortlongname},
11167   sort={\the\glsshorttok},
11168   first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
11169   firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%

11170   text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
11171   plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
11172   description={\protect\glsuserdescription{\the\glslongtok}%
11173     {\the\glslabeltok}}}%
11174   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11175     \csdef{glxtrpostlink\glscategorylabel}{%
11176       \glxtrifwasfirstuse
11177       {%
11178         \glxtruserparen
11179         {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
11180         {\glslabel}%
11181       }%
11182     }%
11183   }%
11184   \glshasattribute{\the\glslabeltok}{regular}%
11185   {%
11186     \glissetattribute{\the\glslabeltok}{regular}{false}%
11187   }%
11188   {}%
11189 }%
11190}%
11191{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11192   \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
11193   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
11194   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
11195   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
11196   \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

11197   \renewcommand*{\glxtrfullformat}[2]{%
11198     \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
11199     \ifglxtrininsertinside\else##2\fi
11200   }%
11201   \renewcommand*{\glxtrfullplformat}[2]{%
11202     \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
11203     \ifglxtrininsertinside\else##2\fi
11204   }%
11205   \renewcommand*{\Glsxtrfullformat}[2]{%
11206     \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
11207     \ifglxtrininsertinside\else##2\fi
11208   }%
11209   \renewcommand*{\Glsxtrfullplformat}[2]{%
11210     \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%

```

```

11211 \ifglxtrinsertinside\else##2\fi
11212 }%

```

In-line format:

```

11213 \renewcommand*{\glxtrinlinefullformat}[2]{%
11214 \glsfirstabbrvuserfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11215 \ifglxtrinsertinside\else##2\fi
11216 \glxtruserparen{\glsfirstlonguserfont{\glssaccesslong{##1}}}{##1}%
11217 }%
11218 \renewcommand*{\glxtrinlinefullplformat}[2]{%
11219 \glsfirstabbrvuserfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11220 \ifglxtrinsertinside\else##2\fi
11221 \glxtruserparen{\glsfirstlonguserfont{\glssaccesslongpl{##1}}}{##1}%
11222 }%
11223 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11224 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11225 \ifglxtrinsertinside\else##2\fi
11226 \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslong{##1}}}{##1}%
11227 }%
11228 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11229 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11230 \ifglxtrinsertinside\else##2\fi
11231 \glxtruserparen{\glsfirstlonguserfont{\Glsaccesslongpl{##1}}}{##1}%
11232 }%
11233 }

```

onguserdesname

```

11234 \newcommand*{\glxtrshortlonguserdesname}{%
11235 \protect\glssabbrvuserfont{\the\glssshorttok}%
11236 \protect\glxtruserparen
11237 {\protect\glslonguserfont{\the\glslongpltok}}}%
11238 {\the\glslabeltok}%
11239 }

```

long-user-desc Like short-postlong-user but leaves the user to specify the description.

```

11240 \newabbreviationstyle{short-postlong-user-desc}%
11241 {%

```

Set accessibility attributes if enabled.

```

11242 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11243 \renewcommand*{\CustomAbbreviationFields}{%
11244 name={\glxtrshortlonguserdesname},
11245 sort={\the\glssshorttok},
11246 first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
11247 firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
11248 text={\protect\glssabbrvuserfont{\the\glssshorttok}},%
11249 plural={\protect\glssabbrvuserfont{\the\glssshortpltok}}}%
11250 }%

```

```

11251 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11252   \csdef{glxstrpostlink\glscategorylabel}{%
11253     \glxstrifwasfirstuse
11254     {%
11255       \glxtruserparen
11256       {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
11257       {\glslabel}%
11258     }%
11259   }%
11260 }%
11261 \glshasattribute{\the\glslabeltok}{regular}%
11262 {%
11263   \glissetattribute{\the\glslabeltok}{regular}{false}%
11264 }%
11265 {}%
11266 }%
11267 }%
11268 {%
11269   \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
11270 }

```

short-user-desc

```

11271 \newabbreviationstyle{long-short-user-desc}%
11272 {%
  Set accessibility attributes if enabled.
11273   \glxstrAccSuppAbbrSetTextShortAttrs\glscategorylabel
  Setup the default fields.
11274   \renewcommand*{\CustomAbbreviationFields}{%
11275     name={\glxtrlongshortuserdescname},
11276     sort={\glxtrlongshortdescsort},%
11277     first={\protect\glsfirstlonguserfont{\the\glslongtok}%
11278       \protect\glxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
11279       {\the\glslabeltok}},%
11280     firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
11281       \protect\glxtruserparen
11282       {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
11283     text={\protect\glsabbrvfont{\the\glsshorttok}},%
11284     plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
11285   }%
  Unset the regular attribute if it has been set.
11286   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11287     \glshasattribute{\the\glslabeltok}{regular}%
11288     {%
11289       \glissetattribute{\the\glslabeltok}{regular}{false}%
11290     }%
11291     {}%
11292   }%

```

```

11293 }%
11294 {%
11295   \GlsXtrUseAbbrStyleFmts{long-short-user}%
11296 }

```

short-long-user

```

11297 \newabbreviationstyle{short-long-user}%
11298 {%

```

Set accessibility attributes if enabled.

```
11299   \glstrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

\glslonguserfont is used in the description since \glsdesc doesn't set the style. (Now in \glsuserdescription.)

```

11300   \renewcommand*{\CustomAbbreviationFields}{%
11301     name={\glstrshortlongname},
11302     sort={\the\glsshorttok},
11303     description={\protect\glsuserdescription{\the\glslongtok}%
11304       {\the\glslabeltok}},%
11305     first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
11306       \protect\glstruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
11307       {\the\glslabeltok}},%
11308     firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
11309       \protect\glstruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
11310       {\the\glslabeltok}},%

11311     text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
11312     plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

11313   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11314     \glshasattribute{\the\glslabeltok}{regular}%
11315     {%
11316       \glissetattribute{\the\glslabeltok}{regular}{false}%
11317     }%
11318   }%
11319 }%
11320 }%
11321 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11322   \renewcommand*{\abbrvpluralsuffix}{\glstrusersuffix}%
11323   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
11324   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
11325   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
11326   \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```
11327   \renewcommand*{\glstrfullformat}[2]{%
```

```

11328 \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11329 \ifglxtrinsertinside\else##2\fi
11330 \glxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
11331 }%
11332 \renewcommand*{\glxtrfullplformat}[2]{%
11333 \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11334 \ifglxtrinsertinside\else##2\fi
11335 \glxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
11336 }%
11337 \renewcommand*{\Glsxtrfullformat}[2]{%
11338 \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
11339 \ifglxtrinsertinside\else##2\fi
11340 \glxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
11341 }%
11342 \renewcommand*{\Glsxtrfullplformat}[2]{%
11343 \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
11344 \ifglxtrinsertinside\else##2\fi
11345 \glxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
11346 }%
11347 }

```

-long-user-desc

```

11348 \newabbreviationstyle{short-long-user-desc}%
11349 {%
    Set accessibility attributes if enabled.
11350 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
    Setup the default fields.
11351 \renewcommand*{\CustomAbbreviationFields}{%
11352   name={\glxtrshortlonguserdescname},
11353   sort={\glxtrshortlongdescsort},%
11354   first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
11355     \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
11356     {\the\glslabeltok}},%
11357   firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
11358     \protect\glxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
11359     {\the\glslabeltok}},%
11360   text={\protect\glsabbrvfont{\the\glsshorttok}},%
11361   plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
11362 }%
    Unset the regular attribute if it has been set.
11363 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11364 \glshasattribute{\the\glslabeltok}{regular}%
11365 {%
11366 \glssetattribute{\the\glslabeltok}{regular}{false}%
11367 }%
11368 {}%
11369 }%

```

```

11370 }%
11371 {%
11372   \GlsXtrUseAbbrStyleFmts{short-long-user}%
11373 }

```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the `keywords` 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 `\glstrlongset` `\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.

```

11374 \newrobustcmd*{\glstrifhyphenstart}[3]{%
11375   \ifx\glsinsert#1\relax
11376     \expandafter\@glstrifhyphenstart#1\relax\relax
11377     \@end@glstrifhyphenstart{#2}{#3}%
11378   \else
11379     \@glstrifhyphenstart#1\relax\relax\@end@glstrifhyphenstart{#2}{#3}%
11380   \fi
11381 }

```

`\trifhyphenstart`

```

11382 \def\@glstrifhyphenstart#1#2\@end@glstrifhyphenstart#3#4{%
11383   \ifx-#1\relax#3\else #4\fi
11384 }

```

`\longhyphenshort`

```
\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.

```
11385 \newcommand*{\glstrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
11386 {%
```

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.

```

11387   \glstrifhyphenstart{#4}{\def\glstrwordsep{-}}{ }%
11388   \glsfirstlonghyphenfont{#2\ifglstrinsertinside{#4}\fi}%
11389   \ifglstrinsertinside\else{#4}\fi
11390   \glstrfullsep{#1}%
11391   \glstrparen{\glsfirstabbrvhyphenfont{#3\ifglstrinsertinside{#4}\fi}%

```



```

11392 \ifglxtrinsertinside\else{#4}\fi}%
11393 }%
11394 }

```

abbrvhyphenfont

```

11395 \newcommand*{\glsabbrvhyphenfont}{\glsabbrvdefaultfont}%

```

abbrvhyphenfont

```

11396 \newcommand*{\glsfirstabbrvhyphenfont}{\glsabbrvhyphenfont}%

```

slonghyphenfont

```

11397 \newcommand*{\glslonghyphenfont}{\glslongdefaultfont}%

```

tlonghyphenfont

```

11398 \newcommand*{\glsfirstlonghyphenfont}{\glslonghyphenfont}%

```

The default short form suffix:

xtrhyphensuffix

```

11399 \newcommand*{\glsxtrhyphensuffix}{\glsxtrabbrvpluralsuffix}

```

en-short-hyphen Designed for use with the markwords attribute.

```

11400 \newabbreviationstyle{long-hyphen-short-hyphen}%
11401 {%

```

Set accessibility attributes if enabled.

```

11402 \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

11403 \renewcommand*{\CustomAbbreviationFields}{%
11404   name={\glsxtrlongshortname},
11405   sort={\the\glsshorttok},
11406   first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
11407     \protect\glsxtrfullsep{\the\glslabeltok}%
11408     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
11409   firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
11410     \protect\glsxtrfullsep{\the\glslabeltok}%
11411     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
11412   text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11413   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
11414   description={\protect\glslonghyphenfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

11415 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11416   \glsasattribute{\the\glslabeltok}{regular}%
11417   {%
11418     \glssetattribute{\the\glslabeltok}{regular}{false}%
11419   }%
11420   {}%
11421 }%

```

```

11422 }%
11423 {%
11424   \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
11425   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
11426   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11427   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11428   \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

11429   \renewcommand*{\glxtrfullformat}[2]{%
11430     \glxtrlonghyphenshort{##1}{\glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
11431   }%
11432   \renewcommand*{\glxtrfullplformat}[2]{%
11433     \glxtrlonghyphenshort{##1}{\glsaccesslongpl{##1}}%
11434     {\glsaccessshortpl{##1}}{##2}%
11435   }%
11436   \renewcommand*{\Glsxtrfullformat}[2]{%
11437     \glxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
11438   }%
11439   \renewcommand*{\Glsxtrfullplformat}[2]{%
11440     \glxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
11441     {\glsaccessshortpl{##1}}{##2}%
11442   }%
11443 }

```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```

11444 \newabbreviationstyle{long-hyphen-short-hyphen-desc}%
11445 {%

```

Set accessibility attributes if enabled.

```

11446   \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11447   \renewcommand*{\CustomAbbreviationFields}{%
11448     name={\glxtrlongshortdescname},
11449     sort={\glxtrlongshortdescsort},
11450     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}}%
11451     \protect\glxtrfullsep{\the\glslabeltok}}%
11452     \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
11453     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}}%
11454     \protect\glxtrfullsep{\the\glslabeltok}}%
11455     \glxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
11456     text={\protect\glsabbrvhyphenfont{\the\glsshorttok}}},%
11457     plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11458   }%

```

Unset the regular attribute if it has been set.

```

11459   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11460     \glshasattribute{\the\glslabeltok}{regular}}%
11461   {%
11462     \glissetattribute{\the\glslabeltok}{regular}{false}}%

```

```

11463 }%
11464 {}%
11465 }%
11466 }%
11467 {%
11468 \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
11469 }

```

nghyphennoshort

```
\glxtrlonghyphennoshort{<label>}{<long>}{<insert>}
```

```
11470 \newcommand*{\glxtrlonghyphennoshort}[3]{%
```

Grouping is needed to localise the redefinitions.

```
11471 {%
```

If *<insert>* starts with a hyphen, redefine `\glxtrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to `\glxtrwordsep` if *<insert>* doesn't start with a hyphen.

```

11472 \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{%
11473 \glslfirstlonghyphenfont{#2\ifglxtrininsertinside{#3}\fi}%
11474 \ifglxtrininsertinside\else{#3}\fi
11475 }%
11476 }

```

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. No accessibility attributes need to be set.

```

11477 \newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
11478 {%
11479 \renewcommand*{\CustomAbbreviationFields}{%
11480 name={\glxtrlongnoshortdescname},
11481 sort={\expandonce\glxtrorglong},
11482 first={\protect\glslfirstlonghyphenfont{\the\glslongtok}},%
11483 firstplural={\protect\glslfirstlonghyphenfont{\the\glslongpltok}},%
11484 text={\protect\glslonghyphenfont{\the\glslongtok}},%
11485 plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
11486 }%

```

Unset the regular attribute if it has been set.

```

11487 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11488 \glshasattribute{\the\glslabeltok}{regular}%
11489 {%
11490 \glissetattribute{\the\glslabeltok}{regular}{false}%
11491 }%
11492 {}%

```

```

11493 }%
11494 }%
11495 {%
11496 \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11497 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
11498 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
11499 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
11500 \renewcommand*\glsfirstlongfont[1]{\glsfirstlonghyphenfont{##1}}%
11501 \renewcommand*\glslongfont[1]{\glslonghyphenfont{##1}}%

```

The format for subsequent use (not used when the regular attribute is set).

```

11502 \renewcommand*\glxtrsubsequentfmt}[2]{%
11503   \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
11504 }%
11505 \renewcommand*\glxtrsubsequentplfmt}[2]{%
11506   \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
11507 }%
11508 \renewcommand*\Glsxtrsubsequentfmt}[2]{%
11509   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
11510 }%
11511 \renewcommand*\Glsxtrsubsequentplfmt}[2]{%
11512   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
11513 }%

```

The inline full form displays the long format followed by the short form in parentheses.

```

11514 \renewcommand*\glxtrinilinefullformat}[2]{%
11515   \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
11516   \glsxtrfullsep{##1}%
11517   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
11518 }%
11519 \renewcommand*\glxtrinilinefullplformat}[2]{%
11520   \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
11521   \glsxtrfullsep{##1}%
11522   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
11523 }%
11524 \renewcommand*\Glsxtrinilinefullformat}[2]{%
11525   \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
11526   \glsxtrfullsep{##1}%
11527   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
11528 }%
11529 \renewcommand*\Glsxtrinilinefullplformat}[2]{%
11530   \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
11531   \glsxtrfullsep{##1}%
11532   \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
11533 }%

```

The first use full form only displays the long form.

```

11534 \renewcommand*\glxtrfullformat}[2]{%
11535   \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%

```

```

11536 }%
11537 \renewcommand*{\glxtrfullplformat}[2]{%
11538   \glxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
11539 }%
11540 \renewcommand*{\Glsxtrfullformat}[2]{%
11541   \glxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
11542 }%
11543 \renewcommand*{\Glsxtrfullplformat}[2]{%
11544   \glxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
11545 }%
11546 }

```

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.

```

11547 \newabbreviationstyle{long-hyphen-noshort-noreg}%
11548 {%

```

Set accessibility attributes if enabled.

```

11549   \glxtrAccSuppAbbrSetNameShortAttrs\glscategorylabel

```

Setup the default fields.

```

11550   \renewcommand*{\CustomAbbreviationFields}{%
11551     name={\glxtrlongnoshortname},
11552     sort={\the\glsshorttok},
11553     first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
11554     firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
11555     text={\protect\glslonghyphenfont{\the\glslongtok}},%
11556     plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
11557     description={\the\glslongtok}%
11558   }%

```

Unset the regular attribute if it has been set.

```

11559   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11560     \glshasattribute{\the\glslabeltok}{regular}%
11561     {%
11562       \glissetattribute{\the\glslabeltok}{regular}{false}%
11563     }%
11564   }%
11565 }%
11566 }%
11567 {%

```

```

11568   \GlsXtrUseAbbrStyleFmts{long-hyphen-noshort-desc-noreg}%
11569 }

```

lsxtrlonghyphen

```
\glxtrlonghyphen{<long>}{<label>}{<insert>}
```

Used by long-hyphen-postshort-hyphen. The <insert> is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```
11570 \newcommand*{\glxtrlonghyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```
11571 {%
11572   \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}%
11573   \glsfirslonghyphenfont{#1}%
11574 }%
11575 }
```

posthyphenshort

```
\glxtrposthyphenshort{<label>}{<insert>}
```

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like \glxtrlonghyphenshort but omits the <long> part. This always uses the singular short form.

```
11576 \newcommand*{\glxtrposthyphenshort}[2]{%
11577 {%
11578   \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
11579   \ifglxtrininsertinside{\glsfirslonghyphenfont{#2}}\else{#2}\fi
11580   \glxtrfullsep{#1}%
11581   \glxtrparen
11582   {\glsfirstabbrvhyphenfont{\glsentryshort{#1}}\ifglxtrininsertinside{#2}\fi}%
11583   \ifglxtrininsertinside\else{#2}\fi
11584 }%
11585 }%
11586 }
```

phensubsequent

```
\glxtrposthyphensubsequent{<label>}{<insert>}
```

Format in the post-link hook for subsequent use. The label is ignored by default.

```
11587 \newcommand*{\glxtrposthyphensubsequent}[2]{%
11588   \glssabrvfont{\ifglxtrininsertinside {#2}\fi}%
11589   \ifglxtrininsertinside \else{#2}\fi
11590 }
```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```
11591 \newabbreviationstyle{long-hyphen-postshort-hyphen}%
11592 {%
```

Set accessibility attributes if enabled.

```
11593 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel
```

Setup the default fields.

```
11594 \renewcommand*{\CustomAbbreviationFields}{%
11595   name={\glxtrlongshortname},
11596   sort={\the\glsshorttok},
11597   first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
11598   firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
11599   text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11600   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
11601   description={\protect\glslonghyphenfont{\the\glslongtok}}}%
11602 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11603   \csdef{glxtrpostlink\glscategorylabel}{%
11604     \glxtrifwasfirstuse
11605     {%
11606       \glxtrposthyphenshort{\glslabel}{\glsinsert}%
11607     }%
11608   }%
```

Put the insertion into the post-link:

```
11609     \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11610   }%
11611 }%
11612 \glshasattribute{\the\glslabeltok}{regular}%
11613 {%
11614   \glssetattribute{\the\glslabeltok}{regular}{false}%
11615 }%
11616 {}%
11617 }%
11618 }%
11619 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
11620 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
11621 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
11622 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11623 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11624 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

Subsequent use needs to omit the insertion:

```
11625 \renewcommand*{\glxtrsubsequentfmt}[2]{%
11626   \glsabbrvfont{\glsaccessshort{##1}}}%
11627 }%
11628 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
11629   \glsabbrvfont{\glsaccessshortpl{##1}}}%
11630 }%
11631 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
11632   \glsabbrvfont{\Glsaccessshort{##1}}}%
11633 }%
11634 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
11635   \Glsabbrvfont{\Glsaccessshortpl{##1}}}%
11636 }
```

```

11635 \glsabbrvfont{\Glsaccessshortpl{##1}}%
11636 }%

```

First use full form:

```

11637 \renewcommand*\glsxtrfullformat}[2]{%
11638 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
11639 }%
11640 \renewcommand*\glsxtrfullplformat}[2]{%
11641 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
11642 }%
11643 \renewcommand*\Glsxtrfullformat}[2]{%
11644 \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
11645 }%
11646 \renewcommand*\Glsxtrfullplformat}[2]{%
11647 \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
11648 }%

```

In-line format.

```

11649 \renewcommand*\glsxtrinlinefullformat}[2]{%
11650 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
11651 \ifglsxtrininsertinside{##2}\fi}%
11652 \ifglsxtrininsertinside \else{##2}\fi
11653 }%
11654 \renewcommand*\glsxtrinlinefullplformat}[2]{%
11655 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
11656 \ifglsxtrininsertinside{##2}\fi}%
11657 \ifglsxtrininsertinside \else{##2}\fi
11658 }%
11659 \renewcommand*\Glsxtrinlinefullformat}[2]{%
11660 \glsfirstlonghyphenfont{\Glsaccesslong{##1}}%
11661 \ifglsxtrininsertinside{##2}\fi}%
11662 \ifglsxtrininsertinside \else{##2}\fi
11663 }%
11664 \renewcommand*\Glsxtrinlinefullplformat}[2]{%
11665 \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}}%
11666 \ifglsxtrininsertinside{##2}\fi}%
11667 \ifglsxtrininsertinside \else{##2}\fi
11668 }%
11669 }

```

`ort-hyphen-desc` Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```

11670 \newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
11671 {%

```

Set accessibility attributes if enabled.

```

11672 \glsxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11673 \renewcommand*\CustomAbbreviationFields}{%
11674 name={\glsxtrlongshortdescname},
11675 sort={\glsxtrlongshortdescsort},%

```



```

11676 first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
11677 firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
11678 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11679 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11680 }%
11681 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11682   \csdef{glsxtrpostlink\glscategorylabel}{%
11683     \glsxtrifwasfirstuse
11684     {%
11685       \glsxtrposthyphenshort{\glslabel}{\glsinsert}}%
11686     }%
11687     {%

```

Put the insertion into the post-link:

```

11688       \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}}%
11689     }%
11690   }%
11691   \glsattribute{\the\glslabeltok}{regular}%
11692   {%
11693     \glssetAttribute{\the\glslabeltok}{regular}{false}%
11694   }%
11695   {}%
11696 }%
11697 }%
11698 {%
11699   \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
11700 }

```

shorthyphenlong

`\glsxtrshorthyphenlong{<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.

```
11701 \newcommand*{\glsxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
11702 {%
```

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.)

```

11703   \glsxtrifhyphenstart{#4}{\def\glsxtrwordsep{-}}{%
11704   \glsfirstabbrvhyphenfont{#2\ifglsxtrininsertinside{#4}\fi}%
11705   \ifglsxtrininsertinside\else{#4}\fi
11706   \glsxtrfullsep{#1}%
11707   \glsxtrparen{\glsfirstlonghyphenfont{#3\ifglsxtrininsertinside{#4}\fi}%
11708   \ifglsxtrininsertinside\else{#4}\fi}%
11709 }%

```

11710 }

hen-long-hyphen Designed for use with the markwords attribute.

11711 \newabbreviationstyle{short-hyphen-long-hyphen}%
11712 {%

Set accessibility attributes if enabled.

11713 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

Setup the default fields.

11714 \renewcommand*{\CustomAbbreviationFields}{%
11715 name={\glxtrshortlongname},
11716 sort={\the\glsshorttok},
11717 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}%
11718 \protect\glxtrfullsep{\the\glslabeltok}}%
11719 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
11720 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}%
11721 \protect\glxtrfullsep{\the\glslabeltok}}%
11722 \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
11723 text={\protect\glssabbrvhyphenfont{\the\glsshorttok}}},%
11724 plural={\protect\glssabbrvhyphenfont{\the\glsshortpltok}}},%
11725 description={\protect\glslonghyphenfont{\the\glslongtok}}}%

Unset the regular attribute if it has been set.

11726 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11727 \glshasattribute{\the\glslabeltok}{regular}}%
11728 {%
11729 \glissetattribute{\the\glslabeltok}{regular}{false}}%
11730 }%
11731 {}%
11732 }%
11733 }%
11734 {%
11735 \renewcommand*{\abbrvpluralsuffix}{\glxtrhyphensuffix}%
11736 \renewcommand*{\glssabbrvfont}[1]{\glssabbrvhyphenfont{##1}}%
11737 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11738 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11739 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

The first use full form and the inline full form are the same for this style.

11740 \renewcommand*{\glxtrfullformat}[2]{%
11741 \glxtrshorthyphenlong{##1}{\glssaccessshort{##1}}{\glssaccesslong{##1}}{##2}}%
11742 }%
11743 \renewcommand*{\glxtrfullplformat}[2]{%
11744 \glxtrshorthyphenlong{##1}%
11745 {\glssaccessshortpl{##1}}{\glssaccesslongpl{##1}}{##2}}%
11746 }%
11747 \renewcommand*{\Glsxtrfullformat}[2]{%
11748 \glxtrshorthyphenlong{##1}{\glssaccessshort{##1}}{\Glsaccesslong{##1}}{##2}}%
11749 }%
11750 \renewcommand*{\Glsxtrfullplformat}[2]{%

```

11751 \glxtrshorthyphenlong{##1}%
11752 {\glssaccessshortpl{##1}}{\glssaccesslongpl{##1}}{##2}%
11753 }%
11754 }

```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```

11755 \newabbreviationstyle{short-hyphen-long-hyphen-desc}%
11756 {%

```

Set accessibility attributes if enabled.

```

11757 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11758 \renewcommand*{\CustomAbbreviationFields}{%
11759   name={\glxtrshortlongdescname},
11760   sort={\glxtrshortlongdescsort},
11761   first={\protect\glsfirstabbrvhyphenfont{\the\glssshorttok}}%
11762   \protect\glxtrfullsep{\the\glslabeltok}}%
11763   \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
11764   firstplural={\protect\glsfirstabbrvhyphenfont{\the\glssshortpltok}}%
11765   \protect\glxtrfullsep{\the\glslabeltok}}%
11766   \glxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
11767   text={\protect\glssabbrvhyphenfont{\the\glssshorttok}}},%
11768   plural={\protect\glssabbrvhyphenfont{\the\glssshortpltok}}}%
11769 }%

```

Unset the regular attribute if it has been set.

```

11770 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11771   \glshasattribute{\the\glslabeltok}{regular}}%
11772   {%
11773     \glsssetAttribute{\the\glslabeltok}{regular}{false}}%
11774   }%
11775   {}%
11776 }%
11777 }%
11778 {%
11779   \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
11780 }

```

sxtrshorthyphen

`\glxtrshorthyphen{<short>}{<label>}{<insert>}`

Used by short-hyphen-postlong-hyphen. The *<insert>* is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```

11781 \newcommand*{\glxtrshorthyphen}[3]{%

```

Grouping is needed to localise the redefinitions.

```

11782 {%

```

```

11783 \glxtrifhyphenstart{#3}{\def\glxtrwordsep{-}}{}%
11784 \glsfirstabbrvhyphenfont{#1}%
11785 }%
11786 }

```

posthyphenlong

`\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.

```

11787 \newcommand*{\glxtrposthyphenlong}[2]{%
11788   {%
11789     \glxtrifhyphenstart{#2}{\def\glxtrwordsep{-}}{}%
11790     \ifglxtrininsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
11791     \glxtrfullsep{#1}%
11792     \glxtrparen
11793     {\glsfirstlonghyphenfont{\glsentrylong{#1}}\ifglxtrininsertinside{#2}\fi}%
11794     \ifglxtrininsertinside\else{#2}\fi
11795   }%
11796 }%
11797 }

```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```

11798 \newabbreviationstyle{short-hyphen-postlong-hyphen}%
11799 {%

```

Set accessibility attributes if enabled.

```

11800 \glxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

11801 \renewcommand*{\CustomAbbreviationFields}{%
11802   name={\glxtrshortlongname},
11803   sort={\the\glsshorttok},
11804   first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11805   firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11806   text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11807   plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
11808   description={\protect\glslonghyphenfont{\the\glslongtok}}}%
11809 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11810   \csdef{glxtrpostlink\glscategorylabel}{%
11811     \glxtrifwasfirstuse
11812     {%
11813       \glxtrposthyphenlong{\glslabel}{\glsinsert}%
11814     }%
11815   }%

```

Put the insertion into the post-link:

```

11816      \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11817    }%
11818  }%
11819  \glshasattribute{\the\glslabeltok}{regular}%
11820  {%
11821    \glsetattribute{\the\glslabeltok}{regular}{false}%
11822  }%
11823  {}%
11824 }%
11825}%
11826{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

11827 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
11828 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
11829 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
11830 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
11831 \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%

```

Subsequent use needs to omit the insertion:

```

11832 \renewcommand*{\glxtrsubsequentfmt}[2]{%
11833   \glsabbrvfont{\glsaccessshort{##1}}%
11834 }%
11835 \renewcommand*{\glxtrsubsequentplfmt}[2]{%
11836   \glsabbrvfont{\glsaccessshortpl{##1}}%
11837 }%
11838 \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
11839   \glsabbrvfont{\Glsaccessshort{##1}}%
11840 }%
11841 \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
11842   \glsabbrvfont{\Glsaccessshortpl{##1}}%
11843 }%

```

First use full form:

```

11844 \renewcommand*{\glxtrfullformat}[2]{%
11845   \glxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
11846 }%
11847 \renewcommand*{\glxtrfullplformat}[2]{%
11848   \glxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
11849 }%
11850 \renewcommand*{\Glsxtrfullformat}[2]{%
11851   \glxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
11852 }%
11853 \renewcommand*{\Glsxtrfullplformat}[2]{%
11854   \glxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
11855 }%

```

In-line format. Commands like `\glxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```

11856 \renewcommand*{\glxtrinlinefullformat}[2]{%

```

```

11857 \glsfirstabbrvhyphenfont{\glsaccessshort{##1}%
11858 \ifglxtrinsertinside{##2}\fi}%
11859 \ifglxtrinsertinside \else{##2}\fi
11860 }%
11861 \renewcommand*{\glxtrinlinefullplformat}[2]{%
11862 \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}%
11863 \ifglxtrinsertinside{##2}\fi}%
11864 \ifglxtrinsertinside \else{##2}\fi
11865 }%
11866 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11867 \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}%
11868 \ifglxtrinsertinside{##2}\fi}%
11869 \ifglxtrinsertinside \else{##2}\fi
11870 }%
11871 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11872 \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}%
11873 \ifglxtrinsertinside{##2}\fi}%
11874 \ifglxtrinsertinside \else{##2}\fi
11875 }%
11876 }

```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```

11877 \newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
11878 {%

```

Set accessibility attributes if enabled.

```
11879 \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel
```

Setup the default fields.

```

11880 \renewcommand*{\CustomAbbreviationFields}{%
11881 name={\glxtrshortlongdescname},
11882 sort={\glxtrshortlongdescsort},%
11883 first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
11884 firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
11885 text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
11886 plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}}%
11887 }%
11888 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11889 \csdef{glxtrpostlink\glscategorylabel}{%
11890 \glxtrifwasfirstuse
11891 {%
11892 \glxtrposthyphenlong{\glslabel}{\glsinsert}%
11893 }%
11894 {%

```

Put the insertion into the post-link:

```

11895 \glxtrposthyphensubsequent{\glslabel}{\glsinsert}%
11896 }%
11897 }%
11898 \glshasattribute{\the\glslabeltok}{regular}%

```

```

11899   {%
11900     \glssetattribute{\the\glslabeltok}{regular}{false}%
11901   }%
11902   {}%
11903   }%
11904 }%
11905 {%
11906   \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
11907 }

```

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`

```

11908 \newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%

```

`stabbrvonlyfont`

```

11909 \newcommand*{\glsfirstabbrvonlyfont}{\glsabbrvonlyfont}%

```

`glslongonlyfont`

```

11910 \newcommand*{\glslongonlyfont}{\glslongdefaultfont}%

```

`rstlongonlyfont`

```

11911 \newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%

```

The default short form suffix:

`lsxtronlysuffix`

```

11912 \newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}

```

`\glsxtronlyname` The default name format for this style.

```

11913 \newcommand*{\glsxtronlyname}{%
11914   \protect\glsabbrvonlyfont{\the\glsshorttok}%
11915 }

```

`only-short-only`

```

11916 \newabbreviationstyle{long-only-short-only}%
11917 {%

```

Set accessibility attributes if enabled.

```

11918   \glsxtrAccSuppAbbrSetFirstLongAttrs\glscategorylabel

```

Setup the default fields.

```

11919   \renewcommand*{\CustomAbbreviationFields}{%
11920     name={\glsxtronlyname},
11921     sort={\the\glsshorttok},
11922     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
11923     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%

```

```

11924 text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
11925 plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
11926 description={\protect\glslongonlyfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

11927 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11928   \glsattribute{\the\glslabeltok}{regular}%
11929   {%
11930     \glssetattribute{\the\glslabeltok}{regular}{false}%
11931   }%
11932   {}%
11933 }%
11934 }%
11935 {%

11936 \renewcommand*{\abbrvpluralsuffix}{\glsxtronlysuffix}%
11937 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
11938 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
11939 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
11940 \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%

```

The first use full form doesn't show the short form.

```

11941 \renewcommand*{\glsxtrfullformat}[2]{%
11942   \glsfirstlongonlyfont{\glsaccesslong{##1}}\ifglsxtrininsertinside##2\fi}%
11943   \ifglsxtrininsertinside\else##2\fi
11944 }%
11945 \renewcommand*{\glsxtrfullplformat}[2]{%
11946   \glsfirstlongonlyfont{\glsaccesslongpl{##1}}\ifglsxtrininsertinside##2\fi}%
11947   \ifglsxtrininsertinside\else##2\fi
11948 }%
11949 \renewcommand*{\Glsxtrfullformat}[2]{%
11950   \glsfirstlongonlyfont{\Glsaccesslong{##1}}\ifglsxtrininsertinside##2\fi}%
11951   \ifglsxtrininsertinside\else##2\fi
11952 }%
11953 \renewcommand*{\Glsxtrfullplformat}[2]{%
11954   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}}\ifglsxtrininsertinside##2\fi}%
11955   \ifglsxtrininsertinside\else##2\fi
11956 }%

```

The inline full form does show the short form.

```

11957 \renewcommand*{\glsxtrininlinefullformat}[2]{%
11958   \glsfirstlongonlyfont{\glsaccesslong{##1}}\ifglsxtrininsertinside##2\fi}%
11959   \ifglsxtrininsertinside\else##2\fi
11960   \glsxtrfullsep{##1}%
11961   \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshort{##1}}}%
11962 }%
11963 \renewcommand*{\glsxtrininlinefullplformat}[2]{%
11964   \glsfirstlongonlyfont{\glsaccesslongpl{##1}}\ifglsxtrininsertinside##2\fi}%
11965   \ifglsxtrininsertinside\else##2\fi
11966   \glsxtrfullsep{##1}%
11967   \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%

```



```

11968 }%
11969 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
11970   \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
11971   \ifglxtrinsertinside\else##2\fi
11972   \glxtrfullsep{##1}%
11973   \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
11974 }%
11975 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
11976   \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
11977   \ifglxtrinsertinside\else##2\fi
11978   \glxtrfullsep{##1}%
11979   \glxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
11980 }%
11981 }

```

xtronlydescsort

```

11982 \newcommand*{\glxtronlydescsort}{\the\glslongtok}

```

xtronlydescname

```

11983 \newcommand*{\glxtronlydescname}{%
11984   \protect\glslongfont{\the\glslongtok}%
11985 }

```

short-only-desc

```

11986 \newabbreviationstyle{long-only-short-only-desc}%
11987 {%

```

Set accessibility attributes if enabled.

```

11988   \glxtrAccSuppAbbrSetTextShortAttrs\glscategorylabel

```

Setup the default fields.

```

11989   \renewcommand*{\CustomAbbreviationFields}{%
11990     name={\glxtronlydescname},
11991     sort={\glxtronlydescsort},%
11992     first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
11993     firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
11994     text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
11995     plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}%
11996   }%

```

Unset the regular attribute if it has been set.

```

11997   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
11998     \glshasattribute{\the\glslabeltok}{regular}%
11999     {%
12000       \glissetattribute{\the\glslabeltok}{regular}{false}%
12001     }%
12002   {}%
12003 }%
12004 }%
12005 {%

```

```

12006 \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
12007 }

```

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 `\glssabrvfont`. 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:

```

12008 \let\@glxtr@org@markright\markright

```

Redefine (grouping not added in case it interferes with the original code):

```

12009 \renewcommand*{\markright}[1]{%
12010 \glxtrmarkhook
12011 \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
12012 \glxtrrestoremarkhook
12013 }

```

`\markboth` Save original definition:

```

12014 \let\@glxtr@org@markboth\markboth

```

Redefine (grouping not added in case it interferes with the original code):

```

12015 \renewcommand*{\markboth}[2]{%
12016 \glxtrmarkhook
12017 \@glxtr@org@markboth
12018 {\@glxtrinmark#1\@glxtrnotinmark}%
12019 {\@glxtrinmark#2\@glxtrnotinmark}%

```

```

12020 \glsxtrrestoremarkhook
12021 }

```

Also do this for \@starttoc

\@starttoc Save original definition:

```

12022 \let\@glsxtr@org@@starttoc\@starttoc

```

Redefine:

```

12023 \renewcommand*{\@starttoc}[1]{%
12024   \glsxtrmarkhook
12025   \@glsxtrinmark
12026   \@glsxtr@org@@starttoc{#1}%
12027   \@glsxtrnotinmark
12028   \glsxtrrestoremarkhook
12029 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```

12030 \newcommand*{\glsxtrRevertMarks}{%
12031   \let\markright\@glsxtr@org@markright
12032   \let\markboth\@glsxtr@org@markboth
12033   \let\@starttoc\@glsxtr@org@@starttoc
12034 }

```

rRevertTocMarks Just restores \@starttoc.

```

12035 \newcommand*{\glsxtrRevertTocMarks}{%
12036   \let\@starttoc\@glsxtr@org@@starttoc
12037 }

```

\glsxtrifinmark

```

12038 \newcommand*{\glsxtrifinmark}[2]{#2}

```

\@glsxtrinmark

```

12039 \newrobustcmd*{\@glsxtrinmark}{%
12040   \let\glsxtrifinmark\@firstoftwo
12041 }

```

glsxtrnotinmark

```

12042 \newrobustcmd*{\@glsxtrnotinmark}{%
12043   \let\glsxtrifinmark\@secondoftwo
12044 }

```

eorpdforheading

```

12045 \ifdef\teorpdfstring
12046 {
12047   \newcommand*{\glsxtrtitleorpdforheading}[3]{\teorpdfstring{#1}{#2}}
12048 }

```

```

12049 {
12050   \newcommand*{\glxstrtitleorpdforheading}[3]{#1}
12051 }

```

\glxstrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```

12052 \newcommand*{\glxstrmarkhook}{%

```

Save current definitions:

```

12053 \let\@glxstr@org@MakeUppercase\MakeUppercase
12054 \let\@glxstr@org@glxstrtitleorpdforheading\glxstrtitleorpdforheading
12055 \let\@glxstr@org@glxstrtitleshort\glxstrtitleshort
12056 \let\@glxstr@org@glxstrtitleshortpl\glxstrtitleshortpl
12057 \let\@glxstr@org@Glsxstrtitleshort\Glsxstrtitleshort
12058 \let\@glxstr@org@Glsxstrtitleshortpl\Glsxstrtitleshortpl
12059 \let\@glxstr@org@glxstrtitlename\glxstrtitlename
12060 \let\@glxstr@org@Glsxstrtitlename\Glsxstrtitlename
12061 \let\@glxstr@org@glxstrtitletext\glxstrtitletext
12062 \let\@glxstr@org@Glsxstrtitletext\Glsxstrtitletext
12063 \let\@glxstr@org@glxstrtitleplural\glxstrtitleplural
12064 \let\@glxstr@org@Glsxstrtitleplural\Glsxstrtitleplural
12065 \let\@glxstr@org@glxstrtitlefirst\glxstrtitlefirst
12066 \let\@glxstr@org@Glsxstrtitlefirst\Glsxstrtitlefirst
12067 \let\@glxstr@org@glxstrtitlefirstplural\glxstrtitlefirstplural
12068 \let\@glxstr@org@Glsxstrtitlefirstplural\Glsxstrtitlefirstplural
12069 \let\@glxstr@org@glxstrtitlelong\glxstrtitlelong
12070 \let\@glxstr@org@glxstrtitlelongpl\glxstrtitlelongpl
12071 \let\@glxstr@org@Glsxstrtitlelong\Glsxstrtitlelong
12072 \let\@glxstr@org@Glsxstrtitlelongpl\Glsxstrtitlelongpl
12073 \let\@glxstr@org@glxstrtitlefull\glxstrtitlefull
12074 \let\@glxstr@org@glxstrtitlefullpl\glxstrtitlefullpl
12075 \let\@glxstr@org@Glsxstrtitlefull\Glsxstrtitlefull
12076 \let\@glxstr@org@Glsxstrtitlefullpl\Glsxstrtitlefullpl

```

New definitions

```

12077 \let\glxstrifinmark\@firstoftwo
12078 \let\MakeUppercase\MakeTextUppercase
12079 \let\glxstrtitleorpdforheading\@thirdofthree
12080 \let\glxstrtitleshort\glxstrheadshort
12081 \let\glxstrtitleshortpl\glxstrheadshortpl
12082 \let\Glsxstrtitleshort\Glsxstrheadshort
12083 \let\Glsxstrtitleshortpl\Glsxstrheadshortpl
12084 \let\glxstrtitlename\glxstrheadname
12085 \let\Glsxstrtitlename\Glsxstrheadname
12086 \let\glxstrtitletext\glxstrheadtext
12087 \let\Glsxstrtitletext\Glsxstrheadtext
12088 \let\glxstrtitleplural\glxstrheadplural
12089 \let\Glsxstrtitleplural\Glsxstrheadplural
12090 \let\glxstrtitlefirst\glxstrheadfirst
12091 \let\Glsxstrtitlefirst\Glsxstrheadfirst

```

```

12092 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
12093 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
12094 \let\glsxtrtitlelong\glsxtrheadlong
12095 \let\glsxtrtitlelongpl\glsxtrheadlongpl
12096 \let\Glsxtrtitlelong\Glsxtrheadlong
12097 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
12098 \let\glsxtrtitlefull\glsxtrheadfull
12099 \let\glsxtrtitlefullpl\glsxtrheadfullpl
12100 \let\Glsxtrtitlefull\Glsxtrheadfull
12101 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
12102 }

```

`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.)

```

12103 \newcommand*{\glsxtrrestoremarkhook}{%
12104   \let\glsxtrinfinmark\@secondoftwo
12105   \let\MakeUppercase\@glsxtr@org@MakeUppercase
12106   \let\glsxtrtitleorpdforheading\@glsxtr@org@glsxtrtitleorpdforheading
12107   \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
12108   \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
12109   \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
12110   \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
12111   \let\glsxtrtitlename\@glsxtr@org@glsxtrtitlename
12112   \let\Glsxtrtitlename\@glsxtr@org@Glsxtrtitlename
12113   \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
12114   \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
12115   \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
12116   \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
12117   \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
12118   \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
12119   \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural
12120   \let\Glsxtrtitlefirstplural\@glsxtr@org@Glsxtrtitlefirstplural
12121   \let\glsxtrtitlelong\@glsxtr@org@glsxtrtitlelong
12122   \let\glsxtrtitlelongpl\@glsxtr@org@glsxtrtitlelongpl
12123   \let\Glsxtrtitlelong\@glsxtr@org@Glsxtrtitlelong
12124   \let\Glsxtrtitlelongpl\@glsxtr@org@Glsxtrtitlelongpl
12125   \let\glsxtrtitlefull\@glsxtr@org@glsxtrtitlefull
12126   \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
12127   \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
12128   \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
12129 }

```

Instead of using one document-wide conditional, use `headuc` attribute to determine whether or not to use the all upper case form.

`glsxtrheadshort` Command used to display short form in the page header.

```

12130 \newcommand*{\glsxtrheadshort}[1]{%

```

```

12131 \protect\NoCaseChange
12132 {%
12133   \glsifattribute{#1}{headuc}{true}%
12134   {%
12135     \GLSxtrshort[noindex,hyper=false]{#1}[]%
12136   }%
12137   {%
12138     \GLSxtrshort[noindex,hyper=false]{#1}[]%
12139   }%
12140 }%
12141 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

12142 \newrobustcmd*{\lsxtrtitleshort}[1]{%
12143   \GLSxtrshort[noindex,hyper=false]{#1}[]%
12144 }

```

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.

```

12145 \newcommand*{\lsxtrheadshortpl}[1]{%
12146   \protect\NoCaseChange
12147   {%
12148     \glsifattribute{#1}{headuc}{true}%
12149     {%
12150       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
12151     }%
12152     {%
12153       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
12154     }%
12155   }%
12156 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents.

```

12157 \newrobustcmd*{\lsxtrtitleshortpl}[1]{%
12158   \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
12159 }

```

GLsxtrheadshort Command used to display short form in the page header with the first letter converted to upper case.

```

12160 \newcommand*{\GLsxtrheadshort}[1]{%
12161   \protect\NoCaseChange
12162   {%
12163     \glsifattribute{#1}{headuc}{true}%
12164     {%
12165       \GLSxtrshort[noindex,hyper=false]{#1}[]%
12166     }%
12167     {%

```

```

12168      \Glsxtrshort[noindex,hyper=false]{#1}[]%
12169    }%
12170  }%
12171 }

```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

12172 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
12173   \Glsxtrshort[noindex,hyper=false]{#1}[]%
12174 }

```

LSxtrtitleshort Command to display short form of abbreviation in section title and table of contents in all upper case.

```

12175 \newrobustcmd*{\GLSxtrtitleshort}[1]{%
12176   \GLSxtrshort[noindex,hyper=false]{#1}[]%
12177 }

```

sxtrheadshortpl Command used to display plural short form in the page header with the first letter converted to upper case.

```

12178 \newcommand*{\Glsxtrheadshortpl}[1]{%
12179   \protect\NoCaseChange
12180   {%
12181     \glsifattribute{#1}{headuc}{true}%
12182     {%
12183       \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
12184     }%
12185     {%
12186       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
12187     }%
12188   }%
12189 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

12190 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
12191   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
12192 }

```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents in all upper case.

```

12193 \newrobustcmd*{\GLSxtrtitleshortpl}[1]{%
12194   \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
12195 }

```

\glsxtrheadname As above but for the name value.

```

12196 \newcommand*{\glsxtrheadname}[1]{%
12197   \protect\NoCaseChange
12198   {%

```

```

12199 \glsifattribute{#1}{headuc}{true}%
12200 {%
12201 \GLSname[noindex,hyper=false]{#1}[]%
12202 }%
12203 {%
12204 \glsname[noindex,hyper=false]{#1}[]%
12205 }%
12206 }%
12207 }

```

`glsxtrtitlename` Command to display name value in section title and table of contents.

```

12208 \newrobustcmd*{\glsxtrtitlename}[1]{%
12209 \glsname[noindex,hyper=false]{#1}[]%
12210 }

```

`\Glsxtrheadname` First letter converted to upper case

```

12211 \newcommand*{\Glsxtrheadname}[1]{%
12212 \protect\NoCaseChange
12213 {%
12214 \glsifattribute{#1}{headuc}{true}%
12215 {%
12216 \GLSname[noindex,hyper=false]{#1}[]%
12217 }%
12218 {%
12219 \Glsname[noindex,hyper=false]{#1}[]%
12220 }%
12221 }%
12222 }

```

`Glsxtrtitlename` Command to display name value in section title and table of contents with the first letter changed to upper case.

```

12223 \newrobustcmd*{\Glsxtrtitlename}[1]{%
12224 \Glsname[noindex,hyper=false]{#1}[]%
12225 }

```

`GLSxtrtitlename` Command to display name value in section title and table of contents in all upper case.

```

12226 \newrobustcmd*{\GLSxtrtitlename}[1]{%
12227 \GLSname[noindex,hyper=false]{#1}[]%
12228 }

```

`\glsxtrheadtext` As above but for the text value.

```

12229 \newcommand*{\glsxtrheadtext}[1]{%
12230 \protect\NoCaseChange
12231 {%
12232 \glsifattribute{#1}{headuc}{true}%
12233 {%
12234 \GLStext[noindex,hyper=false]{#1}[]%
12235 }%

```



```

12236   {%
12237       \glstext[noindex,hyper=false]{#1}[]%
12238   }%
12239 }%
12240 }

```

glsxtrtitletext Command to display text value in section title and table of contents.

```

12241 \newrobustcmd*{\glsxtrtitletext}[1]{%
12242     \glstext[noindex,hyper=false]{#1}[]%
12243 }

```

\Glsxtrheadtext First letter converted to upper case

```

12244 \newcommand*{\Glsxtrheadtext}[1]{%
12245     \protect\NoCaseChange
12246     {%
12247         \glsifattribute{#1}{headuc}{true}%
12248         {%
12249             \GLStext[noindex,hyper=false]{#1}[]%
12250         }%
12251         {%
12252             \Glstext[noindex,hyper=false]{#1}[]%
12253         }%
12254     }%
12255 }

```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```

12256 \newrobustcmd*{\Glsxtrtitletext}[1]{%
12257     \Glstext[noindex,hyper=false]{#1}[]%
12258 }

```

GLSxtrtitletext Command to display text value in section title and table of contents in all upper case.

```

12259 \newrobustcmd*{\GLSxtrtitletext}[1]{%
12260     \GLStext[noindex,hyper=false]{#1}[]%
12261 }

```

lsxtrheadplural As above but for the plural value.

```

12262 \newcommand*{\lsxtrheadplural}[1]{%
12263     \protect\NoCaseChange
12264     {%
12265         \glsifattribute{#1}{headuc}{true}%
12266         {%
12267             \GLSplural[noindex,hyper=false]{#1}[]%
12268         }%
12269         {%
12270             \glsplural[noindex,hyper=false]{#1}[]%
12271         }%
12272     }%
12273 }

```

sxtrtitleplural Command to display plural value in section title and table of contents.

```
12274 \newrobustcmd*{\glsxtrtitleplural}[1]{%
12275   \glsplural[noindex,hyper=false]{#1}[]%
12276 }
```

lsxtrheadplural Convert first letter to upper case.

```
12277 \newcommand*{\Glsxtrheadplural}[1]{%
12278   \protect\NoCaseChange
12279   {%
12280     \glsifattribute{#1}{headuc}{true}%
12281     {%
12282       \GLSplural[noindex,hyper=false]{#1}[]%
12283     }%
12284     {%
12285       \Glsplural[noindex,hyper=false]{#1}[]%
12286     }%
12287   }%
12288 }
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
12289 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
12290   \Glsplural[noindex,hyper=false]{#1}[]%
12291 }
```

Sxtrtitleplural Command to display plural value in section title and table of contents in all upper case.

```
12292 \newrobustcmd*{\GLSxtrtitleplural}[1]{%
12293   \GLSplural[noindex,hyper=false]{#1}[]%
12294 }
```

glsxtrheadfirst As above but for the first value.

```
12295 \newcommand*{\glsxtrheadfirst}[1]{%
12296   \protect\NoCaseChange
12297   {%
12298     \glsifattribute{#1}{headuc}{true}%
12299     {%
12300       \GLSfirst[noindex,hyper=false]{#1}[]%
12301     }%
12302     {%
12303       \glsfirst[noindex,hyper=false]{#1}[]%
12304     }%
12305   }%
12306 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
12307 \newrobustcmd*{\glsxtrtitlefirst}[1]{%
12308   \glsfirst[noindex,hyper=false]{#1}[]%
12309 }
```

Glsxtrheadfirst First letter converted to upper case

```
12310 \newcommand*{\Glsxtrheadfirst}[1]{%
12311   \protect\NoCaseChange
12312   {%
12313     \glsifattribute{#1}{headuc}{true}%
12314     {%
12315       \GLSfirst[noindex,hyper=false]{#1}[]%
12316     }%
12317     {%
12318       \Glsfirst[noindex,hyper=false]{#1}[]%
12319     }%
12320   }%
12321 }
```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```
12322 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
12323   \Glsfirst[noindex,hyper=false]{#1}[]%
12324 }
```

LSxtrtitlefirst Command to display first value in section title and table of contents in all upper case.

```
12325 \newrobustcmd*{\GLSxtrtitlefirst}[1]{%
12326   \GLSfirst[noindex,hyper=false]{#1}[]%
12327 }
```

headfirstplural As above but for the firstplural value.

```
12328 \newcommand*{\glsxtrheadfirstplural}[1]{%
12329   \protect\NoCaseChange
12330   {%
12331     \glsifattribute{#1}{headuc}{true}%
12332     {%
12333       \GLSfirstplural[noindex,hyper=false]{#1}[]%
12334     }%
12335     {%
12336       \glsfirstplural[noindex,hyper=false]{#1}[]%
12337     }%
12338   }%
12339 }
```

itlefirstplural Command to display firstplural value in section title and table of contents.

```
12340 \newrobustcmd*{\glsxtrtitlefirstplural}[1]{%
12341   \glsfirstplural[noindex,hyper=false]{#1}[]%
12342 }
```

headfirstplural First letter converted to upper case

```
12343 \newcommand*{\Glsxtrheadfirstplural}[1]{%
12344   \protect\NoCaseChange
12345   {%
```

```

12346 \glsifattribute{#1}{headuc}{true}%
12347 {%
12348 \GLSfirstplural[noindex,hyper=false]{#1}[]%
12349 }%
12350 {%
12351 \GLSfirstplural[noindex,hyper=false]{#1}[]%
12352 }%
12353 }%
12354 }

```

`\titlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

12355 \newrobustcmd*{\GLSxtrtitlefirstplural}[1]{%
12356 \GLSfirstplural[noindex,hyper=false]{#1}[]%
12357 }

```

`\titlefirstplural` Command to display first value in section title and table of contents in all upper case.

```

12358 \newrobustcmd*{\GLSxtrtitlefirstplural}[1]{%
12359 \GLSfirstplural[noindex,hyper=false]{#1}[]%
12360 }

```

`\glsxtrheadlong` Command used to display long form in the page header.

```

12361 \newcommand*{\glsxtrheadlong}[1]{%
12362 \protect\NoCaseChange
12363 {%
12364 \glsifattribute{#1}{headuc}{true}%
12365 {%
12366 \GLSxtrlong[noindex,hyper=false]{#1}[]%
12367 }%
12368 {%
12369 \glsxtrlong[noindex,hyper=false]{#1}[]%
12370 }%
12371 }%
12372 }

```

`\glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```

12373 \newrobustcmd*{\glsxtrtitlelong}[1]{%
12374 \glsxtrlong[noindex,hyper=false]{#1}[]%
12375 }

```

`\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.

```

12376 \newcommand*{\glsxtrheadlongpl}[1]{%
12377 \protect\NoCaseChange
12378 {%
12379 \glsifattribute{#1}{headuc}{true}%
12380 {%

```

```

12381      \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
12382    }%
12383    {%
12384      \glsxtrlongpl[noindex,hyper=false]{#1}[]%
12385    }%
12386  }%
12387 }

```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents.

```

12388 \newrobustcmd*{\glsxtrtitlelongpl}[1]{%
12389   \glsxtrlongpl[noindex,hyper=false]{#1}[]%
12390 }

```

\Glsxtrheadlong Command used to display long form in the page header with the first letter converted to upper case.

```

12391 \newcommand*{\Glsxtrheadlong}[1]{%
12392   \protect\NoCaseChange
12393   {%
12394     \glsifattribute{#1}{headuc}{true}%
12395     {%
12396       \GLSxtrlong[noindex,hyper=false]{#1}[]%
12397     }%
12398     {%
12399       \Glsxtrlong[noindex,hyper=false]{#1}[]%
12400     }%
12401   }%
12402 }

```

Glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

12403 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
12404   \Glsxtrlong[noindex,hyper=false]{#1}[]%
12405 }

```

GLSxtrtitlelong Command to display long form of abbreviation in section title and table of contents in all upper case.

```

12406 \newrobustcmd*{\GLSxtrtitlelong}[1]{%
12407   \GLSxtrlong[noindex,hyper=false]{#1}[]%
12408 }

```

lSxtrheadlongpl Command used to display plural long form in the page header with the first letter converted to upper case.

```

12409 \newcommand*{\Glsxtrheadlongpl}[1]{%
12410   \protect\NoCaseChange
12411   {%
12412     \glsifattribute{#1}{headuc}{true}%
12413     {%
12414       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%

```

```

12415 }%
12416 {%
12417     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
12418 }%
12419 }%
12420 }

```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

12421 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
12422     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
12423 }

```

`Sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents in all upper case.

```

12424 \newrobustcmd*{\GLSxtrtitlelongpl}[1]{%
12425     \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
12426 }

```

`\glxstrheadfull` Command used to display full form in the page header.

```

12427 \newcommand*{\glxstrheadfull}[1]{%
12428     \protect\NoCaseChange
12429     {%
12430         \gl@ifattribute{#1}{headuc}{true}%
12431         {%
12432             \GLSxtrfull[noindex,hyper=false]{#1}[]%
12433         }%
12434         {%
12435             \glxstrfull[noindex,hyper=false]{#1}[]%
12436         }%
12437     }%
12438 }

```

`glxstrtitlefull` Command to display full form of abbreviation in section title and table of contents.

```

12439 \newrobustcmd*{\glxstrtitlefull}[1]{%
12440     \glxstrfull[noindex,hyper=false]{#1}[]%
12441 }

```

`lxsxtrheadfullpl` 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.

```

12442 \newcommand*{\glxstrheadfullpl}[1]{%
12443     \protect\NoCaseChange
12444     {%
12445         \gl@ifattribute{#1}{headuc}{true}%
12446         {%
12447             \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
12448         }%

```

```

12449   {%
12450       \glsxtrfullpl[noindex,hyper=false]{#1}[]%
12451   }%
12452 }%
12453 }

```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```

12454 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%
12455   \glsxtrfullpl[noindex,hyper=false]{#1}[]%
12456 }

```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```

12457 \newcommand*{\Glsxtrheadfull}[1]{%
12458   \protect\NoCaseChange
12459   {%
12460     \glsifattribute{#1}{headuc}{true}%
12461     {%
12462       \GLSxtrfull[noindex,hyper=false]{#1}[]%
12463     }%
12464     {%
12465       \Glsxtrfull[noindex,hyper=false]{#1}[]%
12466     }%
12467   }%
12468 }

```

Glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

12469 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
12470   \Glsxtrfull[noindex,hyper=false]{#1}[]%
12471 }

```

GLSxtrtitlefull Command to display full form of abbreviation in section title and table of contents in all upper case.

```

12472 \newrobustcmd*{\GLSxtrtitlefull}[1]{%
12473   \GLSxtrfull[noindex,hyper=false]{#1}[]%
12474 }

```

\lsxtrheadfullpl Command used to display plural full form in the page header with the first letter converted to upper case.

```

12475 \newcommand*{\Glsxtrheadfullpl}[1]{%
12476   \protect\NoCaseChange
12477   {%
12478     \glsifattribute{#1}{headuc}{true}%
12479     {%
12480       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
12481     }%
12482     {%

```

```

12483 \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
12484 }%
12485 }%
12486 }

```

`\Sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

12487 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
12488 \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
12489 }

```

`\Sxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents in all upper case.

```

12490 \newrobustcmd*{\GLSxtrtitlefullpl}[1]{%
12491 \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
12492 }

```

`\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.

```

12493 \ifdef\texorpdfstring
12494 {
12495 \newcommand*{\glsfmtshort}[1]{%
12496 \texorpdfstring
12497 {\glsxtrtitleshort{#1}}%
12498 {\glsentryshort{#1}}%
12499 }
12500 }
12501 {
12502 \newcommand*{\glsfmtshort}[1]{%
12503 \glsxtrtitleshort{#1}}
12504 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

12505 \ifdef\texorpdfstring
12506 {
12507 \newcommand*{\glsfmtshortpl}[1]{%
12508 \texorpdfstring
12509 {\glsxtrtitleshortpl{#1}}%
12510 {\glsentryshortpl{#1}}%
12511 }
12512 }
12513 {
12514 \newcommand*{\glsfmtshortpl}[1]{%
12515 \glsxtrtitleshortpl{#1}}
12516 }

```

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).

```
12517 \ifdef\teorpdfstring
12518 {
12519   \newcommand*\Glsfmtshort}[1]{%
12520     \teorpdfstring
12521       {\Glsxtrtitleshort{#1}}%
12522       {\glsentryshort{#1}}%
12523   }
12524 }
12525 {
12526   \newcommand*\Glsfmtshort}[1]{%
12527     \Glsxtrtitleshort{#1}}
12528 }
```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```
12529 \ifdef\teorpdfstring
12530 {
12531   \newcommand*\Glsfmtshortpl}[1]{%
12532     \teorpdfstring
12533       {\Glsxtrtitleshortpl{#1}}%
12534       {\glsentryshortpl{#1}}%
12535   }
12536 }
12537 {
12538   \newcommand*\Glsfmtshortpl}[1]{%
12539     \Glsxtrtitleshortpl{#1}}
12540 }
```

`\glsfmtname` As above but for the name value.

```
12541 \ifdef\teorpdfstring
12542 {
12543   \newcommand*\glsfmtname}[1]{%
12544     \teorpdfstring
12545       {\glsxtrtitlename{#1}}%
12546       {\glsentryname{#1}}%
12547   }
12548 }
12549 {
12550   \newcommand*\glsfmtname}[1]{%
12551     \glsxtrtitlename{#1}}
12552 }
```

`\Glsfmtname` First letter converted to upper case.

```
12553 \ifdef\teorpdfstring
12554 {
12555   \newcommand*\Glsfmtname}[1]{%
12556     \teorpdfstring
12557       {\Glsxtrtitlename{#1}}%
12558       {\glsentryname{#1}}%
12559   }
```

```

12559 }
12560 }
12561 {
12562   \newcommand*{\Glsfmtname}[1]{%
12563     \Glsxtrtitlename{#1}}
12564 }

```

`\Glsfmtname` All upper case.

```

12565 \ifdef\teorpdfstring
12566 {
12567   \newcommand*{\Glsfmtname}[1]{%
12568     \teorpdfstring
12569     {\Glsxtrtitlename{#1}}%
12570     {\glsentryname{#1}}%
12571   }
12572 }
12573 {
12574   \newcommand*{\Glsfmtname}[1]{%
12575     \Glsxtrtitlename{#1}}
12576 }

```

`\glsfmttext` As above but for the text value.

```

12577 \ifdef\teorpdfstring
12578 {
12579   \newcommand*{\glsfmttext}[1]{%
12580     \teorpdfstring
12581     {\glsxtrtitletext{#1}}%
12582     {\glsentrytext{#1}}%
12583   }
12584 }
12585 {
12586   \newcommand*{\glsfmttext}[1]{%
12587     \glsxtrtitletext{#1}}
12588 }

```

`\Glsfmttext` First letter converted to upper case.

```

12589 \ifdef\teorpdfstring
12590 {
12591   \newcommand*{\Glsfmttext}[1]{%
12592     \teorpdfstring
12593     {\Glsxtrtitletext{#1}}%
12594     {\glsentrytext{#1}}%
12595   }
12596 }
12597 {
12598   \newcommand*{\Glsfmttext}[1]{%
12599     \Glsxtrtitletext{#1}}
12600 }

```

`\GLSfmttext` All upper case.

```
12601 \ifdef\texorpdfstring
12602 {
12603   \newcommand*\GLSfmttext}[1]{%
12604     \texorpdfstring
12605     {\GLSxtrtitletext{#1}}%
12606     {\glsentrytext{#1}}%
12607   }
12608 }
12609 {
12610   \newcommand*\GLSfmttext}[1]{%
12611     \GLSxtrtitletext{#1}}
12612 }
```

`\glsfmtplural` As above but for the plural value.

```
12613 \ifdef\texorpdfstring
12614 {
12615   \newcommand*\glsfmtplural}[1]{%
12616     \texorpdfstring
12617     {\glsxtrtitleplural{#1}}%
12618     {\glsentryplural{#1}}%
12619   }
12620 }
12621 {
12622   \newcommand*\glsfmtplural}[1]{%
12623     \glsxtrtitleplural{#1}}
12624 }
```

`\Glsfmtplural` First letter converted to upper case.

```
12625 \ifdef\texorpdfstring
12626 {
12627   \newcommand*\Glsfmtplural}[1]{%
12628     \texorpdfstring
12629     {\Glsxtrtitleplural{#1}}%
12630     {\glsentryplural{#1}}%
12631   }
12632 }
12633 {
12634   \newcommand*\Glsfmtplural}[1]{%
12635     \Glsxtrtitleplural{#1}}
12636 }
```

`\GLSfmtplural` All upper case.

```
12637 \ifdef\texorpdfstring
12638 {
12639   \newcommand*\GLSfmtplural}[1]{%
12640     \texorpdfstring
12641     {\GLSxtrtitleplural{#1}}%
12642     {\glsentryplural{#1}}%
```

```

12643 }
12644 }
12645 {
12646   \newcommand*{\GLSfmtplural}[1]{%
12647     \GLSxtrtitleplural{#1}}
12648 }

```

`\glsfmtfirst` As above but for the first value.

```

12649 \ifdef\texorpdfstring
12650 {
12651   \newcommand*{\glsfmtfirst}[1]{%
12652     \texorpdfstring
12653       {\glsxtrtitlefirst{#1}}%
12654       {\glsentryfirst{#1}}%
12655   }
12656 }
12657 {
12658   \newcommand*{\glsfmtfirst}[1]{%
12659     \glsxtrtitlefirst{#1}}
12660 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

12661 \ifdef\texorpdfstring
12662 {
12663   \newcommand*{\Glsfmtfirst}[1]{%
12664     \texorpdfstring
12665       {\Glsxtrtitlefirst{#1}}%
12666       {\glsentryfirst{#1}}%
12667   }
12668 }
12669 {
12670   \newcommand*{\Glsfmtfirst}[1]{%
12671     \Glsxtrtitlefirst{#1}}
12672 }

```

`\GLSfmtfirst` All upper case.

```

12673 \ifdef\texorpdfstring
12674 {
12675   \newcommand*{\GLSfmtfirst}[1]{%
12676     \texorpdfstring
12677       {\GLSxtrtitlefirst{#1}}%
12678       {\glsentryfirst{#1}}%
12679   }
12680 }
12681 {
12682   \newcommand*{\GLSfmtfirst}[1]{%
12683     \GLSxtrtitlefirst{#1}}
12684 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

12685 \ifdef\txorpdfstring
12686 {
12687   \newcommand*{\glsfmtfirstpl}[1]{%
12688     \txorpdfstring
12689     {\glstrtitlefirstplural{#1}}%
12690     {\glsentryfirstplural{#1}}%
12691   }
12692 }
12693 {
12694   \newcommand*{\glsfmtfirstpl}[1]{%
12695     \glstrtitlefirstplural{#1}}
12696 }
```

`\Glsfmtfirstpl` First letter converted to upper case.

```

12697 \ifdef\txorpdfstring
12698 {
12699   \newcommand*{\Glsfmtfirstpl}[1]{%
12700     \txorpdfstring
12701     {\Glsxtrtitlefirstplural{#1}}%
12702     {\glsentryfirstplural{#1}}%
12703   }
12704 }
12705 {
12706   \newcommand*{\Glsfmtfirstpl}[1]{%
12707     \Glsxtrtitlefirstplural{#1}}
12708 }
```

`\GLSfmtfirstpl` All upper case.

```

12709 \ifdef\txorpdfstring
12710 {
12711   \newcommand*{\GLSfmtfirstpl}[1]{%
12712     \txorpdfstring
12713     {\GLSxtrtitlefirstplural{#1}}%
12714     {\glsentryfirstplural{#1}}%
12715   }
12716 }
12717 {
12718   \newcommand*{\GLSfmtfirstpl}[1]{%
12719     \GLSxtrtitlefirstplural{#1}}
12720 }
```

`\glsfmtlong` As above but for the long value.

```

12721 \ifdef\txorpdfstring
12722 {
12723   \newcommand*{\glsfmtlong}[1]{%
12724     \txorpdfstring
12725     {\glstrtitlelong{#1}}%
12726     {\glsentrylong{#1}}%
```

```

12727 }
12728 }
12729 {
12730   \newcommand*{\glsfmtlong}[1]{%
12731     \glsxtrtitlelong{#1}}
12732 }

```

`\Glsfmtlong` First letter converted to upper case.

```

12733 \ifdef\teorpdfstring
12734 {
12735   \newcommand*{\Glsfmtlong}[1]{%
12736     \teorpdfstring
12737     {\Glsxtrtitlelong{#1}}%
12738     {\glsentrylong{#1}}%
12739   }
12740 }
12741 {
12742   \newcommand*{\Glsfmtlong}[1]{%
12743     \Glsxtrtitlelong{#1}}
12744 }

```

`\GLSfmtlong` All upper case.

```

12745 \ifdef\teorpdfstring
12746 {
12747   \newcommand*{\GLSfmtlong}[1]{%
12748     \teorpdfstring
12749     {\GLSxtrtitlelong{#1}}%
12750     {\glsentrylong{#1}}%
12751   }
12752 }
12753 {
12754   \newcommand*{\GLSfmtlong}[1]{%
12755     \GLSxtrtitlelong{#1}}
12756 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

12757 \ifdef\teorpdfstring
12758 {
12759   \newcommand*{\glsfmtlongpl}[1]{%
12760     \teorpdfstring
12761     {\glsxtrtitlelongpl{#1}}%
12762     {\glsentrylongpl{#1}}%
12763   }
12764 }
12765 {
12766   \newcommand*{\glsfmtlongpl}[1]{%
12767     \glsxtrtitlelongpl{#1}}
12768 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```
12769 \ifdef\txorpdfstring
12770 {
12771   \newcommand*\Glsfmtlongpl}[1]{%
12772     \txorpdfstring
12773     {\Glsxtrtitlelongpl{#1}}%
12774     {\glsentrylongpl{#1}}%
12775   }
12776 }
12777 {
12778   \newcommand*\Glsfmtlongpl}[1]{%
12779     \Glsxtrtitlelongpl{#1}}
12780 }
```

`\GLSfmtlongpl` All upper case.

```
12781 \ifdef\txorpdfstring
12782 {
12783   \newcommand*\GLSfmtlongpl}[1]{%
12784     \txorpdfstring
12785     {\GLSxtrtitlelongpl{#1}}%
12786     {\glsentrylongpl{#1}}%
12787   }
12788 }
12789 {
12790   \newcommand*\GLSfmtlongpl}[1]{%
12791     \GLSxtrtitlelongpl{#1}}
12792 }
```

`\glspdffmtfull` Can't use `\glsxtrinlinefullformat` in PDF bookmarks as it's not fully expandable. This command is for the PDF part of `\txorpdfstring` for the full form.

```
12793 \newcommand*\glspdffmtfull}[1]{\glsentrylong{#1} (\glsentryshort{#1})}%
```

`glspdffmtfullpl` Likewise for plural.

```
12794 \newcommand*\glspdffmtfullpl}[1]{\glsentrylongpl{#1} (\glsentryshortpl{#1})}%
```

`\glsfmtfull` In-line full format.

```
12795 \ifdef\txorpdfstring
12796 {
12797   \newcommand*\glsfmtfull}[1]{%
12798     \txorpdfstring
12799     {\glsxtrtitlefull{#1}}%
12800     {\glspdffmtfull{#1}}%
12801   }
12802 }
12803 {
12804   \newcommand*\glsfmtfull}[1]{%
12805     \glsxtrtitlefull{#1}}
12806 }
```

`\Glsfmtfull` First letter converted to upper case.

```
12807 \ifdef\texorpdfstring
12808 {
12809   \newcommand*\Glsfmtfull}[1]{%
12810     \texorpdfstring
12811     {\Glsxtrtitlefull{#1}}%
12812     {\glspdffmtfull{#1}{}}%
12813   }
12814 }
12815 {
12816   \newcommand*\Glsfmtfull}[1]{%
12817     \Glsxtrtitlefull{#1}}
12818 }
```

`\GLSfmtfull` All upper case.

```
12819 \ifdef\texorpdfstring
12820 {
12821   \newcommand*\GLSfmtfull}[1]{%
12822     \texorpdfstring
12823     {\GLSxtrtitlefull{#1}}%
12824     {\glspdffmtfull{#1}}%
12825   }
12826 }
12827 {
12828   \newcommand*\GLSfmtfull}[1]{%
12829     \GLSxtrtitlefull{#1}}
12830 }
```

`\glsfmtfullpl` In-line full plural format.

```
12831 \ifdef\texorpdfstring
12832 {
12833   \newcommand*\glsfmtfullpl}[1]{%
12834     \texorpdfstring
12835     {\glsxtrtitlefullpl{#1}}%
12836     {\glspdffmtfullpl{#1}}%
12837   }
12838 }
12839 {
12840   \newcommand*\glsfmtfullpl}[1]{%
12841     \glsxtrtitlefullpl{#1}}
12842 }
```

`\Glsfmtfullpl` First letter converted to upper case.

```
12843 \ifdef\texorpdfstring
12844 {
12845   \newcommand*\Glsfmtfullpl}[1]{%
12846     \texorpdfstring
12847     {\Glsxtrtitlefullpl{#1}}%
12848     {\glspdffmtfullpl{#1}{}}%
12849   }
```



```

12849 }
12850 }
12851 {
12852   \newcommand*{\GLSfmtfullpl}[1]{%
12853     \GLSxtrtitlefullpl{#1}}
12854 }

```

`\GLSfmtfullpl` All upper case.

```

12855 \ifdef\teorpdfstring
12856 {
12857   \newcommand*{\GLSfmtfullpl}[1]{%
12858     \teorpdfstring
12859     {\GLSxtrtitlefullpl{#1}}%
12860     {\glspdfdfmtfullpl{#1}{}}%
12861   }
12862 }
12863 {
12864   \newcommand*{\GLSfmtfullpl}[1]{%
12865     \GLSxtrtitlefullpl{#1}}
12866 }

```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

12867 \newcommand*{\RequireGlossariesExtraLang}[1]{%
12868   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
12869 }

```

`sariesExtraLang`

```

12870 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
12871   \ProvidesFile{glossariesxtr-#1.ldf}%
12872 }

```

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.

```

12873 \newcommand{\glxtr@loaddialect}{%
12874   \IfTrackedLanguageFileExists{\this@dialect}%
12875   {glossariesxtr-}% prefix
12876   {.ldf}%
12877   {%

```

```

12878 \RequireGlossariesExtraLang{\CurrentTrackedTag}%
12879 }%
12880 {}% not found

```

If glossaries-extra-bib2gls has been loaded, \@glstrdialecthook will check for the associated script, otherwise it will do nothing.

```

12881 \@glstrdialecthook
12882 }

```

```

12883 \@ifpackageloaded{tracklang}
12884 {%
12885 \AnyTrackedLanguages
12886 {%
12887 \ForEachTrackedDialect{\this@dialect}{\glstr@loaddialect}%
12888 }%
12889 {}%
12890 }
12891 {}

```

Load glossaries-extra-stylemods if required.

```

12892 \@glstr@redefstyles
    and set the style:
12893 \@glstr@do@style

```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for bib2gls and is automatically loaded by the record option.

```

12894 \NeedsTeXFormat{LaTeX2e}
12895 \ProvidesPackage{glossaries-extra-bib2gls}[2020/02/28 v1.43 (NLCT)]

```

Provide convenient shortcut commands for predefined glossary types.

printunsrtacronyms

```

12896 \ifglssacronym
12897 \providecommand*\printunsrtacronyms[1][]{%
12898 \printunsrtglossary[type=\acronymtype,#1]}%
12899 \fi

```

printunsrtindex

```

12900 \ifglossaryexists{index}
12901 {
12902 \providecommand*\printunsrtindex[1][]{%
12903 \printunsrtglossary[type=index,#1]}%
12904 }{}

```

printunsrtsymbols

```

12905 \ifglossaryexists{symbols}

```

```

12906 {
12907   \providecommand*\printunsrtsymbols}[1] [] {%
12908     \printunsrtglossary[type=symbols,#1]}%
12909 }{}

```

intunsrtnumbers

```

12910 \ifglossaryexists{numbers}
12911 {
12912   \providecommand*\printunsrtnumbers}[1] [] {%
12913     \printunsrtglossary[type=numbers,#1]}%
12914 }{}

```

rtaabbreviations

```

12915 \ifglossaryexists{abbreviations}
12916 {
12917   \providecommand*\printunsrtaabbreviations}[1] [] {%
12918     \printunsrtglossary[type=abbreviations,#1]}%
12919 }{}

```

splaynumberlist Allow \glsdisplaynumberlist and make it robust.

```

12920 \renewcommand*\glsdisplaynumberlist}[1] {%
12921   \glsdoifexists{#1}%
12922   {%
12923     {\let\bibglsgdelimN\glsnumlistsep
12924       \let\bibglslastDelimN\glsnumlistlastsep
12925       \glstrusefield{#1}{location}}%
12926   }%
12927   }%
12928 }
12929 \robustify\glsdisplaynumberlist

```

entrynumberlist

```

12930 \renewcommand*\glsentrynumberlist}[1]{\glstrusefield{#1}{location}}

```

These are some convenient macros for use with custom rules.

\glsheX

```

12931 \newcommand*\glsheX{\string\u}

```

lscapturedgroup

```

12932 \newcommand*\glsapturedgroup{\string\$}

```

nZeroChildCount For use with bib2gls's save-child-count resource option.

```

12933 \newcommand*\GlsXtrIfHasNonZeroChildCount}[3] {%
12934   \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
12935 }

```

`rprovidecommand` For use in @preamble, this behaves like `\providecommand` in the document but like `\renewcommand` in bib2gls.

```
12936 \newcommand*{\glstrprovidecommand}{\providecommand}
```

`glstrenewcommand` Like `\renewcommand` but only generates a warning rather than an error if the command isn't defined.

```
12937 \newcommand*{\glstrenewcommand}{\@star@or@long\glstr@renewcommand}
```

`tr@renewcommand`

```
12938 \newcommand*{\glstr@renewcommand}[1]{%
12939   \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}\endgroup
12940   \expandafter\@ifundefined\@gtempa
12941     {%
12942       \GlossariesExtraWarning{can't redefine \noexpand#1(not already defined)}%
12943     }%
12944     \relax
12945     \relax
12946     \let\@ifdefinable\@rc@ifdefinable
12947     \new@command#1%
12948 }
```

`lossarylocation` For use with indexcounter and bib2gls.

```
12949 \newcommand*{\glstr@wrglossarylocation}[2]{#1}
```

`indexCounterLink`

`\GlsXtrIndexCounterLink{<text>}{<label>}`

For use with indexcounter and bib2gls.

```
12950 \ifdef\hyperref
12951 {%
12952   \newcommand*{\GlsXtrIndexCounterLink}[2]{%
12953     \glstrifhasfield{indexcounter}{#2}%
12954     {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
12955     {#1}%
12956   }
12957 }
12958 {
12959   \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
12960 }
```

`GlsXtrDualField`

`\GlsXtrDualField`

The internal field used to store the dual label. The dual-field defaults to dual if no value is supplied so that's used as the default.

```
12961 \newcommand*{\GlsXtrDualField}{dual}
```

XtrDualBackLink

```
\GlsXtrDualBackLink{<text>}{<label>}
```

Adds a hyperlink to the dual entry.

```
12962 \newcommand*{\GlsXtrDualBackLink}[2]{%
12963   \glsxtrifhasfield{\GlsXtrDualField}{#2}%
12964   {\gls hyperlink[#1]{\gls currentfieldvalue}}%
12965   {#2}%
12966 }
```

TeXEntryAliases Convenient shortcut for use with entry-type-aliases to alias standard BibTeX entry types to @bibtexentry.

```
12967 \newcommand*{\GlsXtrBibTeXEntryAliases}{%
12968   article=bibtexentry,
12969   book=bibtexentry,
12970   booklet=bibtexentry,
12971   conference=bibtexentry,
12972   inbook=bibtexentry,
12973   incollection=bibtexentry,
12974   inproceedings=bibtexentry,
12975   manual=bibtexentry,
12976   mastersthesis=bibtexentry,
12977   misc=bibtexentry,
12978   phdthesis=bibtexentry,
12979   proceedings=bibtexentry,
12980   techreport=bibtexentry,
12981   unpublished=bibtexentry
12982 }
```

ideBibTeXFields Convenient shortcut to define the standard BibTeX fields.

```
12983 \newcommand*{\GlsXtrProvideBibTeXFields}{%
12984   \glsaddstoragekey{address}{\glsxtrbibaddress}%
12985   \glsaddstoragekey{author}{\glsxtrbibauthor}%
12986   \glsaddstoragekey{booktitle}{\glsxtrbibbooktitle}%
12987   \glsaddstoragekey{chapter}{\glsxtrbibchapter}%
12988   \glsaddstoragekey{edition}{\glsxtrbibedition}%
12989   \glsaddstoragekey{howpublished}{\glsxtrbibhowpublished}%
12990   \glsaddstoragekey{institution}{\glsxtrbibinstitution}%
12991   \glsaddstoragekey{journal}{\glsxtrbibjournal}%
12992   \glsaddstoragekey{month}{\glsxtrbibmonth}%
12993   \glsaddstoragekey{note}{\glsxtrbibnote}%
12994   \glsaddstoragekey{number}{\glsxtrbibnumber}%
12995   \glsaddstoragekey{organization}{\glsxtrbiborganization}%
12996   \glsaddstoragekey{pages}{\glsxtrbibpages}%

```

```

12997 \glsaddstoragekey{publisher}{\glsxtrbibpublisher}%
12998 \glsaddstoragekey{school}{\glsxtrbibschoo}%
12999 \glsaddstoragekey{series}{\glsxtrbibseries}%
13000 \glsaddstoragekey{title}{\glsxtrbibtitle}%
13001 \glsaddstoragekey{bibtextype}{\glsxtrbibtype}%
13002 \glsaddstoragekey{volume}{\glsxtrbibvolume}%
13003 }

```

Multiple supplementary references are only supported with bib2gls.

`\glstisuppllocation` This is like `\glsxtrsupphypernumber` but the second argument is the external file name (which isn't obtained from the `externallocation` attribute). The third argument is the formatting (encap) control sequence *name*. This is ignored by default, but is set by bib2gls to the original encap in case it's required.

```

13004 \newcommand*\glsxtrmultisuppllocation}[3]{%
13005   {%
13006     \def\glsxtrsuppllocationurl{#2}%
13007     \glshypernumber{#1}%
13008   }%
13009 }

```

`\glsrdisplaysupplloc`

`\glsxtrdisplaysupplloc{<prefix>}{<counter>}{<format>}{<src>}{<location>}`

This is like `\glsnoidxdisplayloc` but is used for supplementary locations and so requires an extra argument.

```

13010 \newcommand*\glsxtrdisplaysupplloc[5]{%
13011   \setentrycounter{#1}{#2}%
13012   \glsxtrmultisuppllocation{#5}{#4}{#3}%
13013 }

```

`\glsplaylocnameref` `\glsxtrdisplaylocnameref{<prefix>}{<counter>}{<format>}{<location>}{<name>}{<href>}{<hcounter>}{<external file>}` Used with the [nameref] record package option. The `<href>` argument was obtained from `\@currentHref` and the `<hcounter>` argument was obtained from `\theHentrycounter`, which is more reliable. If hyperref hasn't been loaded, this just behaves like `\glsnoidxdisplayloc`.

```

13014 \ifundef\hyperlink
13015 {
13016   \newcommand*\glsxtrdisplaylocnameref[8]{%
13017     \glsnoidxdisplayloc{#1}{#2}{#3}{#4}%
13018   }
13019 }
13020 {

```

Default action uses `<hcounter>`. Equations and pages typically don't have a title, so check the counter name (otherwise the title may section or chapter title, which may be confusing). As from v1.42, this now checks if the control sequence `\glsxtr<counter>locfmt` is defined.

```

13021 \newcommand*{\glxtrdisplaylocnameref}[8]{%
13022   \ifcsdef{glxtr#2locfmt}%
13023     {\glxtrnameref link{#3}{\csuse{glxtr#2locfmt}{#4}{#5}}{#2.#7}{#8}}%
13024     {%
13025       \ifstrempy{#5}%
13026       {%

```

No title, so just use the location as the link text.

```

13027     \glxtrnameref link{#3}{#4}{#2.#7}{#8}%
13028   }%
13029   {%
13030     \ifstrequal{#2}{page}%
13031     {\glxtrnameref link{#3}{#4}{#2.#7}{#8}}%
13032     {\glxtrnameref link{#3}{#5}{#2.#7}{#8}}%
13033   }%
13034 }%
13035 }
13036 }

```

re equationlocfmt

```
\glxstre equationlocfmt{<location>}{<title>}
```

```
13037 \newcommand*{\glxstre equationlocfmt}[2]{( #1 )}
```

sxtrnameref link

```
\glxstrfmt nameref link{<format>}{<title>}{<href>}{<external file>}
```

```
13038 \newcommand*{\glxtrnameref link}[4]{%
```

Locally change \glshypernumber to \@firstofone to remove the normal location hyperlink.

```

13039 \begingroup
13040 \let\glshypernumber \@firstofone

```

If the *<external file>* argument is empty, an internal link is used, otherwise an external one is needed.

```

13041 \ifstrempy{#4}%
13042 {\glxstrfmt internalnameref{#3}{#1}{#2}}%
13043 {\glxstrfmt externalnameref{#3}{#1}{#2}{#4}}%
13044 \endgroup
13045 }

```

sxtrnameloclink

```
\glstrnamerefloclink{<prefix>}{<counter>}{<format>}{<location>}{<text>}{<external file>}
```

Like `\@gls@numberlink`, this creates a hyperlink to the target obtained from the prefix, counter and location but uses `<text>` as the hyperlink text. As with regular indexing, this will fail if the target name can't be formed by prefixing the location value.

```
13046 \newcommand{\glstrnameloclink}[6]{%
13047   \begingroup
13048   \setentrycounter[#1]{#2}%
13049   \def\glstr@locationhypertext{#5}%
13050   \let\glshypernumber\@firstofone
13051   \def\@glsnumberformat{#3}%
13052   \def\glstrsupplocationurl{#6}%
13053   \toks@={}%
13054   \@glstr@bibgls@removespaces#4 \@nil
13055   \endgroup
13056 }
```

`ls@removespaces`

```
13057 \def\@glstr@bibgls@removespaces#1 #2\@nil{%
13058   \toks@=\expandafter{\the\toks@#1}%
13059   \ifx\#2\%
13060     \edef\x{\the\toks@}%
13061     \ifx\x\empty
13062       \else
13063         \protected@edef\x{\glstentrycounter\@gls@counterprefix\the\toks@}%
13064         \ifdefined\glstrsupplocationurl
13065           {%
13066             \expandafter\glstrfmtinternalnameref\expandafter{\x}%
13067             {\@glsnumberformat}{\glstr@locationhypertext}%
13068           }%
13069           {%
13070             \expandafter\glstrfmtexternalnameref\expandafter{\x}%
13071             {\@glsnumberformat}{\glstr@locationhypertext}{\glstrsupplocationurl}%
13072           }%
13073         \fi
13074       \else
13075         \@gls@ReturnAfterFi{%
13076           \@glstr@bibgls@removespaces#2\@nil
13077         }%
13078       \fi
13079 }
```

`internalnameref`

```
\glstrfmtinternalnameloc{<target>}{<format>}{<title>}
```



```

13080 \newcommand*{\glxtrfmtinternalnameref}[3]{%
13081   \csuse{#2}{\glsdohyperlink{#1}{#3}}%
13082 }

```

externalnameref

```
\glxtrfmtexternalnameloc{<target>}{<format>}{<title>}{<file>}
```

```

13083 \newcommand*{\glxtrfmtexternalnameref}[4]{%
13084   \csuse{#2}{\hyperref{#4}{#1}{#3}}%
13085 }

```

glxtrSetWidest

```
\glxtrSetWidest{<type>}{<level>}{<text>}
```

As from **bib2gls** v1.8, this is used by the set-widest resource option for the alttree and the styles provided by the glossary-longextra package.

```
13086 \newcommand*{\glxtrSetWidest}[3]{%
```

Check which style options have been provided. (The style packages may not have been loaded.)

```

13087   \ifdef\glupdatewidest
13088   {%
13089     \ifdef\glslongextraUpdateWidest
13090     {%

```

Relevant style packages all loaded. If the *<type>* has been given, append to glossary preamble.

```

13091     \ifstrempy{#1}
13092     {%
13093       \glupdatewidest[#2]{#3}%
13094       \ifnum#2=0\relax
13095         \glslongextraUpdateWidest{#3}%
13096       \else
13097         \glslongextraUpdateWidestChild{#2}{#3}%
13098       \fi
13099     }%
13100     {%
13101       \apptoglossarypreamble[#1]{\glupdatewidest[#2]{#3}}%
13102       \ifnum#2=0\relax
13103         \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
13104       \else
13105         \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
13106       \fi
13107     }%
13108   }%
13109   {%

```

Only altree.

```

13110     \ifstrempy{#1}
13111     {%
13112         \glsupdatewidest[#2]{#3}%
13113     }%
13114     {%
13115         \apptoglossarypreamble[#1]{\glsupdatewidest[#2]{#3}}%
13116     }%
13117 }%
13118 }%
13119 {%

```

\glsupdatewidest hasn't been defined. This could just mean that the glossaries-extra-stylemods package hasn't been loaded.

```

13120     \ifdef\glssetwidest
13121     {%
13122         \ifdef\glslongextraUpdateWidest
13123         {%

```

Relevant glossary-tree and glossary-longextra have been loaded. If the *<type>* has been given, append to glossary preamble.

```

13124     \ifstrempy{#1}
13125     {%
13126         \glssetwidest[#2]{#3}%
13127         \ifnum#2=0\relax
13128             \glslongextraUpdateWidest{#3}%
13129         \else
13130             \glslongextraUpdateWidestChild{#2}{#3}%
13131         \fi
13132     }%
13133     {%
13134         \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
13135         \ifnum#2=0\relax
13136             \apptoglossarypreamble[#1]{\glslongextraUpdateWidest{#3}}%
13137         \else
13138             \apptoglossarypreamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
13139         \fi
13140     }%
13141 }%
13142 {%

```

Only altree.

```

13143     \ifstrempy{#1}
13144     {%
13145         \glssetwidest[#2]{#3}%
13146     }%
13147     {%
13148         \apptoglossarypreamble[#1]{\glssetwidest[#2]{#3}}%
13149     }%
13150 }%

```

```

13151 }%
13152 {%
13153     \ifdef\glslongextraUpdateWidest
13154     {%
        glossary-longextra has been loaded.
13155         \ifstrempy{#1}
13156         {%
13157             \ifnum#2=0\relax
13158                 \glslongextraUpdateWidest{#3}%
13159             \else
13160                 \glslongextraUpdateWidestChild{#2}{#3}%
13161             \fi
13162         }%
13163     {%
13164         \ifnum#2=0\relax
13165             \apptoglossary preamble[#1]{\glslongextraUpdateWidest{#3}}%
13166         \else
13167             \apptoglossary preamble[#1]{\glslongextraUpdateWidestChild{#2}{#3}}%
13168         \fi
13169     }%
13170 }%
        Neither glossary-tree nor glossary-longextra have been loaded. Do nothing.
13171 }%
13172 }%
13173 }%
13174 }

```

tWidestFallback

```
\glxtrSetWidestFallback{<max depth>}{<list>}
```

Used when **bib2gls** can't determine the widest name. The *<list>* argument is a comma-separated list of glossary labels. The *<max depth>* refers to the maximum hierarchical depth. This will either be 0 (only top-level entries) or 2 (up to two child-levels).

```

13175 \newcommand*{\glxtrSetWidestFallback}[2]{%
13176     \ifnum#1=0\relax
13177     \ifdef\glsFindWidestTopLevelName
13178     {%
13179         \glsFindWidestTopLevelName[#2]%
13180     }%
13181     {%
13182         \GlossariesExtraWarning{You need stylemods={tree} to
13183             provide a fallback for set-widest}%
13184     }%
13185     \else
13186     \ifdef\glsFindWidestLevelTwo

```

```

13187  {%
13188    \glsFindWidestLevelTwo[#2]%
13189    \ifdef\glslongextraUpdateWidestChild
13190    {%
13191      \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnamei}}%
13192      \glslongextraUpdateWidestChild{#1}{\csuse{@glswidestnameii}}%
13193    }%
13194    {}%
13195  }%
13196  {%
13197    \GlossariesExtraWarning{You need stylemods={tree} to
13198      provide a fallback for set-widest}%
13199  }%
13200  \fi
13201 }

```

`r@labelprefixes` List of label prefixes.

```

13202 \newcommand*{\@glsxtr@labelprefixes}{}

```

`arlabelprefixes` List of label prefixes.

```

13203 \newcommand*{\glsxtrclearlabelprefixes}{%
13204   \renewcommand*{\@glsxtr@labelprefixes}{}%
13205 }

```

`raddlabelprefix` Add prefix to the list. These should be added in the order of precedence with the last one as a fallback. This doesn't check against duplicates as it may be useful to replicate a prefix at the end as the fallback.

```

13206 \newcommand*{\glsxtraddlabelprefix}[1]{%
13207   \ifstrempy{#1}%
13208   {\glsxtraddlabelprefix{\empty}}%
13209   {%
13210     \ifdefempty\@glsxtr@labelprefixes
13211     {\def\@glsxtr@labelprefixes{#1}}%
13212     {\appto\@glsxtr@labelprefixes{,#1}}%
13213   }%
13214 }

```

`pendlabelprefix` Inserts at the start of the list.

```

13215 \newcommand*{\glsxtrprependlabelprefix}[1]{%
13216   \ifstrempy{#1}%
13217   {\glsxtrprependlabelprefix{\empty}}%
13218   {%
13219     \ifdefempty\@glsxtr@labelprefixes
13220     {\def\@glsxtr@labelprefixes{#1}}%
13221     {\pretto\@glsxtr@labelprefixes{#1,}}%
13222   }%
13223 }

```

labelprefixlist

```
\glxstrifinlabelprefixlist{<prefix>}{<true>}{<false>}
```

Test if the given prefix is in the list.

```
13224 \newcommand*{\glxstrifinlabelprefixlist}[3]{%
13225   \ifstrempy{#1}%
13226   {\glxstrifinlabelprefixlist{\empty}{#2}{#3}}%
13227   {%
13228     \DTLifinlist{#1}{\@glxstr@labelprefixes}{#2}{#3}%
13229   }%
13230 }
```

prefixlabellist This is provided for the benefit of **bib2gls**. It's possible that the user may add more prefixes after the start of the document, but that can lead to inconsistencies. The final element of the list (the fallback) is the only prefix of interest for **bib2gls**.

```
13231 \AtBeginDocument{%
13232   \protected@write\@auxout{}{\string\providecommand{\string\@glxstr@prefixlabellist}[1]{}}%
13233   \protected@write\@auxout{}{\string\@glxstr@prefixlabellist{\@glxstr@labelprefixes}}%
13234 }
```

t@prefixedlabel Iterate through all the prefixes and find the first prefix and label combination that exists. If none found, this could mean that it's the first \TeX run, so the last prefix in the list needs to be the fallback one. Grouping is used in case of a nested for loop.

```
13235 \newcommand*{\@glxstr@get@prefixedlabel}[1]{%
13236   \begingroup
```

Initialise to the unprefixed label in the event that the list is empty.

```
13237   \edef\@gls@thislabel{#1}%
13238   \@for\@glxstr@prefix:=\@glxstr@labelprefixes\do
13239   {%
13240     \edef\@gls@thislabel{\@glxstr@prefix#1}%
13241     \ifglstryexists{\@gls@thislabel}{\@endfortrue}{}%
13242   }%
13243   \edef\x{\endgroup\noexpand\def\noexpand\@gls@thislabel{\@gls@thislabel}}\x
13244 }
```

\dglS Like \gls but tries the prefixes. (Can't use \pgls as that's provided by glossaries-prefix.) Since this command is designed for bib2gls's dual entry system, the "d" stands for "dual".

```
13245 \newrobustcmd*{\dglS}{\@gls@hyp@opt\dglS}
```

\@dglS

```
13246 \newcommand*{\@dglS}[2][{}]{%
13247   \@glxstr@get@prefixedlabel{#2}%
13248   \new@ifnextchar[{\@gls@{#1}{\@gls@thislabel}}{\@gls@{#1}{\@gls@thislabel}}[]}%
13249 }
```

```

\dglspl
13250 \newrobustcmd*{\dglspl}{\@gls@hyp@opt\dglspl}

\@dglspl
13251 \newcommand*{\@dglspl}[2][]{%
13252   \@glsxtr@get@prefixedlabel{#2}%
13253   \new@ifnextchar[{\@glspl@{#1}{\@gls@thislabel}}{\@glspl@{#1}{\@gls@thislabel}}[]}%
13254 }

\dGls
13255 \newrobustcmd*{\dGls}{\@gls@hyp@opt\dGls}

\@dGls
13256 \newcommand*{\@dGls}[2][]{%
13257   \@glsxtr@get@prefixedlabel{#2}%
13258   \new@ifnextchar[{\@Gls@{#1}{\@gls@thislabel}}{\@Gls@{#1}{\@gls@thislabel}}[]}%
13259 }

\dGlspl
13260 \newrobustcmd*{\dGlspl}{\@gls@hyp@opt\dGlspl}

\@dGlspl
13261 \newcommand*{\@dGlspl}[2][]{%
13262   \@glsxtr@get@prefixedlabel{#2}%
13263   \new@ifnextchar[{\@Glspl@{#1}{\@gls@thislabel}}{\@Glspl@{#1}{\@gls@thislabel}}[]}%
13264 }

\dGLS
13265 \newrobustcmd*{\dGLS}{\@gls@hyp@opt\dGLS}

\@dGLS
13266 \newcommand*{\@dGLS}[2][]{%
13267   \@glsxtr@get@prefixedlabel{#2}%
13268   \new@ifnextchar[{\@GLS@{#1}{\@gls@thislabel}}{\@GLS@{#1}{\@gls@thislabel}}[]}%
13269 }

\dGLSpl
13270 \newrobustcmd*{\dGLSpl}{\@gls@hyp@opt\dGLSpl}

\@dGLSpl
13271 \newcommand*{\@dGLSpl}[2][]{%
13272   \@glsxtr@get@prefixedlabel{#2}%
13273   \new@ifnextchar[{\@GLSpl@{#1}{\@gls@thislabel}}{\@GLSpl@{#1}{\@gls@thislabel}}[]}%
13274 }

```

`\dglslink` Like `\glslink` but tries the prefixes.

```
13275 \newrobustcmd*{\dglslink}[3][\]{%
13276   \@glsxtr@get@prefixedlabel{#2}%
13277   \glslink[#1]{\@gls@thislabel}{#3}%
13278 }
```

`\dglsdisp` Like `\glsdisp` but tries the prefixes.

```
13279 \newrobustcmd*{\dglsdisp}[3][\]{%
13280   \@glsxtr@get@prefixedlabel{#2}%
13281   \glsdisp[#1]{\@gls@thislabel}{#3}%
13282 }
```

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.

`\Alpha`

```
13283 \providecommand*{\Alpha}{\mathrm{A}}
```

`\Beta`

```
13284 \providecommand*{\Beta}{\mathrm{B}}
```

`\Epsilon`

```
13285 \providecommand*{\Epsilon}{\mathrm{E}}
```

`\Zeta`

```
13286 \providecommand*{\Zeta}{\mathrm{Z}}
```

`\Eta`

```
13287 \providecommand*{\Eta}{\mathrm{H}}
```

`\Iota`

```
13288 \providecommand*{\Iota}{\mathrm{I}}
```

`\Kappa`

```
13289 \providecommand*{\Kappa}{\mathrm{K}}
```

`\Mu`

```
13290 \providecommand*{\Mu}{\mathrm{M}}
```

`\Nu`

```
13291 \providecommand*{\Nu}{\mathrm{N}}
```

```

\Omicron
13292 \providecommand*\Omicron{\mathrm{O}}

\Rho
13293 \providecommand*\Rho{\mathrm{P}}

\Tau
13294 \providecommand*\Tau{\mathrm{T}}

\Chi
13295 \providecommand*\Chi{\mathrm{X}}

\Digamma
13296 \providecommand*\Digamma{\mathrm{F}}

\omicron
13297 \providecommand*\omicron{\mathit{o}}

        Provide corresponding upright characters if upgreek has been loaded. (The upper case
        characters are the same as above.)
13298 \@ifpackageloaded{upgreek}%
13299 {

\Upalpha
13300 \providecommand*\Upalpha{\mathrm{A}}

\Upbeta
13301 \providecommand*\Upbeta{\mathrm{B}}

\Upsilon
13302 \providecommand*\Upsilon{\mathrm{E}}

\Upzeta
13303 \providecommand*\Upzeta{\mathrm{Z}}

\Upeta
13304 \providecommand*\Upeta{\mathrm{H}}

\Upiota
13305 \providecommand*\Upiota{\mathrm{I}}

\Upkappa
13306 \providecommand*\Upkappa{\mathrm{K}}

\Upmu
13307 \providecommand*\Upmu{\mathrm{M}}

```



```

\Upnu
13308 \providecommand*\Upnu{\mathrm{N}}

\Upomicron
13309 \providecommand*\Upomicron{\mathrm{O}}

\Uprho
13310 \providecommand*\Uprho{\mathrm{P}}

\Uptau
13311 \providecommand*\Uptau{\mathrm{T}}

\Upchi
13312 \providecommand*\Upchi{\mathrm{X}}

\upomicron
13313 \providecommand*\upomicron{\mathrm{o}}

13314 }%
13315 {}% 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={\glsxtrcontrolrules
; \glsxtrspacerules
; \glsxtrnonprintablerules
; \glsxtrcombiningdiacriticrules
, \glsxtrhyphenrules
< \glsxtrgeneralpuncrules
< \glsxtrdigitrules
< \glsxtrfractionrules
< \glsxtrGeneralLatinIVrules
< \glsxtrMathItalicGreekIrules
}

```

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.

```

13316 \newcommand*{\glxtrcontrolrules}{%
13317 \string'\glshex 200B\string'\string=\glshex 200C\string=\glshex 200D
13318 \string=\glshex 200E\string=\glshex 200F\string=\glshex 0000\string=\glshex 0001
13319 \string=\glshex 0002\string=\glshex 0003\string=\glshex 0004\string=\glshex 0005
13320 \string=\glshex 0006\string=\glshex 0007\string=\glshex 0008
13321 \string=\string'\glshex 0009\string'\string=\string'\glshex 000B\string'
13322 \string=\glshex 000E\string=\glshex 000F\string=\string'\glshex
13323 0010\string'\string=\glshex 0011
13324 \string=\glshex 0012\string=\glshex 0013\string=\glshex 0014\string=\glshex 0015
13325 \string=\glshex 0016\string=\glshex 0017\string=\glshex 0018\string=\glshex 0019
13326 \string=\glshex 001A\string=\glshex 001B\string=\glshex 001C\string=\glshex 001D
13327 \string=\glshex 001E\string=\glshex 001F\string=\glshex 007F\string=\glshex 0080
13328 \string=\glshex 0081\string=\glshex 0082\string=\glshex 0083\string=\glshex 0084
13329 \string=\glshex 0085\string=\glshex 0086\string=\glshex 0087\string=\glshex 0088
13330 \string=\glshex 0089\string=\glshex 008A\string=\glshex 008B\string=\glshex 008C
13331 \string=\glshex 008D\string=\glshex 008E\string=\glshex 008F\string=\glshex 0090
13332 \string=\glshex 0091\string=\glshex 0092\string=\glshex 0093\string=\glshex 0094
13333 \string=\glshex 0095\string=\glshex 0096\string=\glshex 0097\string=\glshex 0098
13334 \string=\glshex 0099\string=\glshex 009A\string=\glshex 009B\string=\glshex 009C
13335 \string=\glshex 009D\string=\glshex 009E\string=\glshex 009F
13336 }

```

lsxtrspacerules These are space characters.

```

13337 \newcommand*{\glxtrspacerules}{%
13338 \string' \string'\string;
13339 \string'\glshex 00A0\string'\string;
13340 \string'\glshex 2000\string'\string;
13341 \string'\glshex 2001\string'\string;
13342 \string'\glshex 2002\string'\string;
13343 \string'\glshex 2003\string'\string;
13344 \string'\glshex 2004\string'\string;
13345 \string'\glshex 2005\string'\string;
13346 \string'\glshex 2006\string'\string;
13347 \string'\glshex 2007\string'\string;
13348 \string'\glshex 2008\string'\string;
13349 \string'\glshex 2009\string'\string;
13350 \string'\glshex 200A\string'\string;
13351 \string'\glshex 3000\string'
13352 }

```

nprintablerules These are non-printable characters (BOM, tabs, line feed and carriage return).

```

13353 \newcommand*{\glxtrnonprintablerules}{%
13354 \string'\glshex FEFF\string'\string;
13355 \string'\glshex 000A\string'\string;
13356 \string'\glshex 0009\string'\string;

```

```

13357 \string'\glshex 000C\string'\string;
13358 \string'\glshex 000B\string'
13359 }

```

gdiacriticrules Combining diacritic marks. This is split into multiple macros.

```

13360 \newcommand*{\glxtrcombiningdiacriticrules}{%
13361 \glxtrcombiningdiacriticIrules\string;
13362 \glxtrcombiningdiacriticIIrules\string;
13363 \glxtrcombiningdiacriticIIIrules\string;
13364 \glxtrcombiningdiacriticIVrules
13365 }

```

diacriticIrules First set of combining diacritic marks.

```

13366 \newcommand*{\glxtrcombiningdiacriticIrules}{%
13367 \glshex 0301\string;% combining acute
13368 \glshex 0300\string;% combining grave
13369 \glshex 0306\string;% combining breve
13370 \glshex 0302\string;% combining circumflex
13371 \glshex 030C\string;% combining caron
13372 \glshex 030A\string;% combining ring
13373 \glshex 030D\string;% combining vertical line above
13374 \glshex 0308\string;% combining diaeresis
13375 \glshex 030B\string;% combining double acute
13376 \glshex 0303\string;% combining tilde
13377 \glshex 0307\string;% combining dot above
13378 \glshex 0304% combining macron
13379 }

```

diacriticIIrules Second set of combining diacritic marks.

```

13380 \newcommand*{\glxtrcombiningdiacriticIIrules}{%
13381 \glshex 0337\string;% combining short solidus overlay
13382 \glshex 0327\string;% combining cedilla
13383 \glshex 0328\string;% combining ogonek
13384 \glshex 0323\string;% combining dot below
13385 \glshex 0332\string;% combining low line
13386 \glshex 0305\string;% combining overline
13387 \glshex 0309\string;% combining hook above
13388 \glshex 030E\string;% combining double vertical line above
13389 \glshex 030F\string;% combining double grave accent
13390 \glshex 0310\string;% combining candrabindu
13391 \glshex 0311\string;% combining inverted breve
13392 \glshex 0312\string;% combining turned comma above
13393 \glshex 0313\string;% combining comma above
13394 \glshex 0314\string;% combining reversed comma above
13395 \glshex 0315\string;% combining comma above right
13396 \glshex 0316\string;% combining grave accent below
13397 \glshex 0317% combining acute accent below
13398 }

```

acriticIIIrules Third set of combining diacritic marks.

```
13399 \newcommand*{\glxtrcombiningdiacriticIIIrules}{%
13400 \glshex 0318\string;% combining left tack below
13401 \glshex 0319\string;% combining right tack below
13402 \glshex 031A\string;% combining left angle above
13403 \glshex 031B\string;% combining horn
13404 \glshex 031C\string;% combining left half ring below
13405 \glshex 031D\string;% combining up tack below
13406 \glshex 031E\string;% combining down tack below
13407 \glshex 031F\string;% combining plus sign below
13408 \glshex 0320\string;% combining minus sign below
13409 \glshex 0321\string;% combining palatalized hook below
13410 \glshex 0322\string;% combining retroflex hook below
13411 \glshex 0324\string;% combining diaeresis below
13412 \glshex 0325\string;% combining ring below
13413 \glshex 0326\string;% combining comma below
13414 \glshex 0329\string;% combining vertical line below
13415 \glshex 032A\string;% combining bridge below
13416 \glshex 032B\string;% combining inverted double arch below
13417 \glshex 032C\string;% combining caron below
13418 \glshex 032D\string;% combining circumflex accent below
13419 \glshex 032E\string;% combining breve below
13420 \glshex 032F\string;% combining inverted breve below
13421 \glshex 0330\string;% combining tilde below
13422 \glshex 0331\string;% combining macron below
13423 \glshex 0333\string;% combining double low line
13424 \glshex 0334\string;% combining tilde overlay
13425 \glshex 0335\string;% combining short stroke overlay
13426 \glshex 0336\string;% combining long stroke overlay
13427 \glshex 0338\string;% combining long solidus overlay
13428 \glshex 0339\string;% combining combining right half ring below
13429 \glshex 033A\string;% combining inverted bridge below
13430 \glshex 033B\string;% combining square below
13431 \glshex 033C\string;% combining seagull below
13432 \glshex 033D\string;% combining x above
13433 \glshex 033E\string;% combining vertical tilde
13434 \glshex 033F\string;% combining double overline
13435 \glshex 0342\string;% combining Greek perispomeni
13436 \glshex 0344\string;% combining Greek dialytika tonos
13437 \glshex 0345\string;% combining Greek ypogegrammeni
13438 \glshex 0360\string;% combining double tilde
13439 \glshex 0361\string;% combining double inverted breve
13440 \glshex 0483\string;% combining Cyrillic titlo
13441 \glshex 0484\string;% combining Cyrillic palatalization
13442 \glshex 0485\string;% combining Cyrillic dasia pneumata
13443 \glshex 0486% combining Cyrillic psili pneumata
13444 }
```

iacriticIVrules Fourth set of combining diacritic marks.

```

13445 \newcommand*{\glxtrcombiningdiacriticIVrules}{%
13446 \glshex 20D0\string;% combining left harpoon above
13447 \glshex 20D1\string;% combining right harpoon above
13448 \glshex 20D2\string;% combining long vertical line overlay
13449 \glshex 20D3\string;% combining short vertical line overlay
13450 \glshex 20D4\string;% combining anticlockwise arrow above
13451 \glshex 20D5\string;% combining clockwise arrow above
13452 \glshex 20D6\string;% combining left arrow above
13453 \glshex 20D7\string;% combining right arrow above
13454 \glshex 20D8\string;% combining ring overlay
13455 \glshex 20D9\string;% combining clockwise ring overlay
13456 \glshex 20DA\string;% combining anticlockwise ring overlay
13457 \glshex 20DB\string;% combining three dots above
13458 \glshex 20DC\string;% combining four dots above
13459 \glshex 20DD\string;% combining enclosing circle
13460 \glshex 20DE\string;% combining enclosing square
13461 \glshex 20DF\string;% combining enclosing diamond
13462 \glshex 20E0\string;% combining enclosing circle backslash
13463 \glshex 20E1% combining left right arrow above
13464 }

```

sxtrhyphenrules Hyphens.

```

13465 \newcommand*{\glxtrhyphenrules}{%
13466 \string'\string-\string'\string;% ASCII hyphen
13467 \glshex 00AD\string;% soft hyphen
13468 \glshex 2010\string;% hyphen
13469 \glshex 2011\string;% non-breaking hyphen
13470 \glshex 2012\string;% figure dash
13471 \glshex 2013\string;% en dash
13472 \glshex 2014\string;% em dash
13473 \glshex 2015\string;% horizontal bar
13474 \glshex 2212\string=\glshex 207B\string=\glshex 208B% minus sign
13475 }

```

eneralpuncrules General punctuation.

```

13476 \newcommand*{\glxtrgeneralpuncrules}{%
13477 \glxtrgeneralpuncIrules
13478 \string<\glxtrcurrencyrules
13479 \string<\glxtrgeneralpuncIIrules
13480 }

```

eneralpuncIrules First set of general punctuation.

```

13481 \newcommand*{\glxtrgeneralpuncIrules}{%
13482 \string'\glshex 005F\string'% underscore
13483 \string<\glshex 00AF% macron
13484 \string<\string'\glshex 002C\string'% comma
13485 \string<\string'\glshex 003B\string'% semi-colon
13486 \string<\string'\glshex 003A\string'% colon
13487 \string<\string'\glshex 0021\string'% exclamation mark

```

13488 \string<\glshex 00A1% inverted exclamation mark
 13489 \string<\string'\glshex 003F\string'% question mark
 13490 \string<\glshex 00BF% inverted question mark
 13491 \string<\string'\glshex 002F\string'% solidus
 13492 \string<\string'\glshex 002E\string'% full stop
 13493 \string<\glshex 00B4% acute accent
 13494 \string<\string'\glshex 0060\string'% grave accent
 13495 \string<\string'\glshex 005E\string'% circumflex accent
 13496 \string<\glshex 00A8% diaeresis
 13497 \string<\string'\glshex 007E\string'% tilde
 13498 \string<\glshex 00B7% middle dot
 13499 \string<\glshex 00B8% cedilla
 13500 \string<\string'\glshex 0027\string'% straight apostrophe
 13501 \string<\string'\glshex 0022\string'% straight double quote
 13502 \string<\glshex 00AB% left guillemet
 13503 \string<\glshex 00BB% right guillemet
 13504 \string<\string'\glshex 0028\string'% left parenthesis
 13505 \string=<\glshex 207D\string=<\glshex 208D% super/subscript left parenthesis
 13506 \string<\string'\glshex 0029\string'% right parenthesis
 13507 \string=<\glshex 207E\string=<\glshex 208E% super/subscript right parenthesis
 13508 \string<\string'\glshex 005B\string'% left square bracket
 13509 \string<\string'\glshex 005D\string'% right square bracket
 13510 \string<\string'\glshex 007B\string'% left curly bracket
 13511 \string<\string'\glshex 007D\string'% right curly bracket
 13512 \string<\glshex 00A7% section sign
 13513 \string<\glshex 00B6% pilcrow sign
 13514 \string<\glshex 00A9% copyright sign
 13515 \string<\glshex 00AE% registered sign
 13516 \string<\string'\glshex 0040\string'% at sign
 13517 }

trcurrencyrules General punctuation.

13518 \newcommand*{\glxtrcurrencyrules}{%
 13519 \glshex 00A4% currency sign
 13520 \string<\glshex 0E3F% Thai currency symbol baht
 13521 \string<\glshex 00A2% cent sign
 13522 \string<\glshex 20A1% colon sign
 13523 \string<\glshex 20A2% cruzeiro sign
 13524 \string<\string'\glshex 0024\string'% dollar sign
 13525 \string<\glshex 20AB% dong sign
 13526 \string<\glshex 20AC% euro sign
 13527 \string<\glshex 20A3% French franc sign
 13528 \string<\glshex 20A4% lira sign
 13529 \string<\glshex 20A5% mill sign
 13530 \string<\glshex 20A6% naira sign
 13531 \string<\glshex 20A7% peseta sign
 13532 \string<\glshex 00A3% pound sign
 13533 \string<\glshex 20A8% rupee sign
 13534 \string<\glshex 20AA% new sheqel sign

```

13535 \string<\glshex 20A9% won sign
13536 \string<\glshex 00A5% yen sign
13537 }

```

eralpuncIIrules Second set of general punctuation.

```

13538 \newcommand*{\glxtrgeneralpuncIIrules}{%
13539 \string'\glshex 002A\string'% asterisk
13540 \string<\string'\glshex 005C\string'% backslash
13541 \string<\string'\glshex 0026\string'% ampersand
13542 \string<\string'\glshex 0023\string'% hash sign
13543 \string<\string'\glshex 0025\string'% percent sign
13544 \string<\string'\glshex 002B\string'% plus sign
13545 \string=\glshex 207A\string=\glshex 208A% super/subscript plus sign
13546 \string<\glshex 00B1% plus-minus sign
13547 \string<\glshex 00F7% division sign
13548 \string<\glshex 00D7% multiplication sign
13549 \string<\string'\glshex 003C\string'% less-than sign
13550 \string<\string'\glshex 003D\string'% equals sign
13551 \string<\string'\glshex 003E\string'% greater-than sign
13552 \string<\glshex 00AC% not sign
13553 \string<\string'\glshex 007C\string'% vertical bar (pipe)
13554 \string<\glshex 00A6% broken bar
13555 \string<\glshex 00B0% degree sign
13556 \string<\glshex 00B5% micron sign
13557 }

```

eralLatinIrules Basic Latin alphabet.

```

13558 \newcommand*{\glxtrGeneralLatinIrules}{%
13559 \glxtrLatinA
13560 \string<b,B%
13561 \string<c,C%
13562 \string<d,D%
13563 \string<\glxtrLatinE
13564 \string<f,F%
13565 \string<g,G%
13566 \string<\glxtrLatinH
13567 \string<\glxtrLatinI
13568 \string<j,J%
13569 \string<\glxtrLatinK
13570 \string<\glxtrLatinL
13571 \string<\glxtrLatinM
13572 \string<\glxtrLatinN
13573 \string<\glxtrLatinO
13574 \string<\glxtrLatinP
13575 \string<q,Q%
13576 \string<r,R%
13577 \string<\glxtrLatinS
13578 \string<\glxtrLatinT
13579 \string<u,U%

```

```

13580 \string<v,V%
13581 \string<w,W%
13582 \string<\glxtrLatinX
13583 \string<y,Y%
13584 \string<z,Z
13585 }

```

allLatinIIrules General Latin alphabet (eth between D and E, ß treated as SS).

```

13586 \newcommand*{\glxtrGeneralLatinIIrules}{%
13587 \glxtrLatinA
13588 \string<b,B%
13589 \string<c,C%
13590 \string<d,D%
13591 \string<\glxtrLatinEth
13592 \string<\glxtrLatinE
13593 \string<f,F%
13594 \string<g,G%
13595 \string<\glxtrLatinH
13596 \string<\glxtrLatinI
13597 \string<j,J%
13598 \string<\glxtrLatinK
13599 \string<\glxtrLatinL
13600 \string<\glxtrLatinM
13601 \string<\glxtrLatinN
13602 \string<\glxtrLatinO
13603 \string<\glxtrLatinP
13604 \string<q,Q%
13605 \string<r,R%
13606 \string<\glxtrLatinS
13607 \string& SS \string, \glxtrLatinEszettSs
13608 \string<\glxtrLatinT
13609 \string<u,U%
13610 \string<v,V%
13611 \string<w,W%
13612 \string<\glxtrLatinX
13613 \string<y,Y%
13614 \string<z,Z%
13615 }

```

allLatinIIIrules General Latin alphabet (eth between D and E, ß treated as SZ).

```

13616 \newcommand*{\glxtrGeneralLatinIIIrules}{%
13617 \glxtrLatinA
13618 \string<b,B%
13619 \string<c,C%
13620 \string<d,D%
13621 \string<\glxtrLatinEth
13622 \string<\glxtrLatinE
13623 \string<f,F%
13624 \string<g,G%

```



```

13625 \string<\glxtrLatinH
13626 \string<\glxtrLatinI
13627 \string<j,J%
13628 \string<\glxtrLatinK
13629 \string<\glxtrLatinL
13630 \string<\glxtrLatinM
13631 \string<\glxtrLatinN
13632 \string<\glxtrLatinO
13633 \string<\glxtrLatinP
13634 \string<q,Q%
13635 \string<r,R%
13636 \string<\glxtrLatinS
13637 \string& SZ, \glxtrLatinEszettSz
13638 \string<\glxtrLatinT
13639 \string<u,U%
13640 \string<v,V%
13641 \string<w,W%
13642 \string<\glxtrLatinX
13643 \string<y,Y%
13644 \string<z,Z%
13645 }

```

`\glxtrGeneralLatinIVrules` General Latin alphabet (Æ treated as AE and Ætreated as OE, Þtreated as TH, ß treated as SS, eth between D and E).

```

13646 \newcommand*{\glxtrGeneralLatinIVrules}{%
13647 \glxtrLatinA
13648 \string& AE , \glxtrLatinAELigature
13649 \string<b,B%
13650 \string<c,C%
13651 \string<d,D%
13652 \string<\glxtrLatinEth
13653 \string<\glxtrLatinE
13654 \string<f,F%
13655 \string<g,G%
13656 \string<\glxtrLatinH
13657 \string<\glxtrLatinI
13658 \string<j,J%
13659 \string<\glxtrLatinK
13660 \string<\glxtrLatinL
13661 \string<\glxtrLatinM
13662 \string<\glxtrLatinN
13663 \string<\glxtrLatinO
13664 \string& OE , \glxtrLatinOELigature
13665 \string<\glxtrLatinP
13666 \string<q,Q%
13667 \string<r,R%
13668 \string<\glxtrLatinS
13669 \string& SS , \glxtrLatinEszettSs
13670 \string<\glxtrLatinT

```

```

13671 \string& th =\glshex 00DE
13672 \string& TH =\glshex 00FE
13673 \string<u,U%
13674 \string<v,V%
13675 \string<w,W%
13676 \string<\glsxtrLatinX
13677 \string<y,Y%
13678 \string<z,Z%
13679 }

```

eralLatinVrules General Latin alphabet (eth between D and E, ß treated as SS, Þ treated as TH).

```

13680 \newcommand*{\glsxtrGeneralLatinVrules}{%
13681 \glsxtrLatinA
13682 \string<b,B%
13683 \string<c,C%
13684 \string<d,D%
13685 \string<\glsxtrLatinEth
13686 \string<\glsxtrLatinE
13687 \string<f,F%
13688 \string<g,G%
13689 \string<\glsxtrLatinH
13690 \string<\glsxtrLatinI
13691 \string<j,J%
13692 \string<\glsxtrLatinK
13693 \string<\glsxtrLatinL
13694 \string<\glsxtrLatinM
13695 \string<\glsxtrLatinN
13696 \string<\glsxtrLatinO
13697 \string<\glsxtrLatinP
13698 \string<q,Q%
13699 \string<r,R%
13700 \string<\glsxtrLatinS
13701 \string& SS , \glsxtrLatinEszettSs
13702 \string<\glsxtrLatinT
13703 \string& th =\glshex 00DE
13704 \string& TH =\glshex 00FE
13705 \string<u,U%
13706 \string<v,V%
13707 \string<w,W%
13708 \string<\glsxtrLatinX
13709 \string<y,Y%
13710 \string<z,Z%
13711 }

```

ralLatinVrules General Latin alphabet (eth between D and E, ß treated as SZ, Þ treated as TH).

```

13712 \newcommand*{\glsxtrGeneralLatinVrules}{%
13713 \glsxtrLatinA
13714 \string<b,B%
13715 \string<c,C%

```

```

13716 \string<d,D%
13717 \string<\glxtrLatinEth
13718 \string<\glxtrLatinE
13719 \string<f,F%
13720 \string<g,G%
13721 \string<\glxtrLatinH
13722 \string<\glxtrLatinI
13723 \string<j,J%
13724 \string<\glxtrLatinK
13725 \string<\glxtrLatinL
13726 \string<\glxtrLatinM
13727 \string<\glxtrLatinN
13728 \string<\glxtrLatinO
13729 \string<\glxtrLatinP
13730 \string<q,Q%
13731 \string<r,R%
13732 \string<\glxtrLatinS
13733 \string& SZ , \glxtrLatinEszettSz
13734 \string<\glxtrLatinT
13735 \string& th =\glshex 00DE
13736 \string& TH =\glshex 00FE
13737 \string<u,U%
13738 \string<v,V%
13739 \string<w,W%
13740 \string<\glxtrLatinX
13741 \string<y,Y%
13742 \string<z,Z%
13743 }

```

allLatinVIIrules 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).

```

13744 \newcommand*{\glxtrGeneralLatinVIIrules}{%
13745 \glxtrLatinA
13746 \string<\glxtrLatinAELigature
13747 \string<b,B%
13748 \string<c,C%
13749 \string<d,D%
13750 \string<\glxtrLatinEth
13751 \string<\glxtrLatinE
13752 \string<f,F%
13753 \string<\glxtrLatinInsularG
13754 \string<\glxtrLatinH
13755 \string<\glxtrLatinI
13756 \string<j,J%
13757 \string<\glxtrLatinK
13758 \string<\glxtrLatinL
13759 \string<\glxtrLatinM
13760 \string<\glxtrLatinN
13761 \string<\glxtrLatinO

```

```

13762 \string<\glxtrLatinOELigature
13763 \string<\glxtrLatinP
13764 \string<q,Q%
13765 \string<r,R%
13766 \string<\glshex 017F=\glxtrLatinS % s and long s
13767 \string<\glxtrLatinT
13768 \string<\glxtrLatinThorn
13769 \string<u,U%
13770 \string<v,V%
13771 \string< w\string=\glshex 01BF, W\string=\glshex 01F7
13772 \string<\glxtrLatinX
13773 \string<y,Y%
13774 \string<z,Z%
13775 }

```

1LatinVIIIrules 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).

```

13776 \newcommand*{\glxtrGeneralLatinVIIIrules}{%
13777 \glxtrLatinA
13778 \string& AE , \glxtrLatinAELigature
13779 \string<b,B%
13780 \string<c,C%
13781 \string<\glshex 00F0\string;d,\glshex 00D0\string;D% D and eth
13782 \string<\glxtrLatinE
13783 \string<f,F%
13784 \string<g,G%
13785 \string<\glxtrLatinH
13786 \string<\glxtrLatinI
13787 \string<j,J%
13788 \string<\glxtrLatinK
13789 \string<\glshex 0142\string=\glxtrLatinL\string=\glshex 0141% L and \L
13790 \string<\glxtrLatinM
13791 \string<\glxtrLatinN
13792 \string<\glshex 00F8\string=\glxtrLatinO\string=\glshex 00D8% O and \O
13793 \string& OE , \glxtrLatinOELigature
13794 \string<\glxtrLatinP
13795 \string<q,Q%
13796 \string<r,R%
13797 \string<\glxtrLatinS
13798 \string& SS , \glxtrLatinEszettSs
13799 \string<\glxtrLatinT
13800 \string& th =\glshex 00DE
13801 \string& TH =\glshex 00FE
13802 \string<u,U%
13803 \string<v,V%
13804 \string<w,W%
13805 \string<\glxtrLatinX
13806 \string<y,Y%
13807 \string<z,Z%

```

13808 }

\glsxtrLatinA

13809 \newcommand*{\glsxtrLatinA}{%
13810 a\string=\glshex 00AA\string=\glshex 2090,A
13811 }

\glsxtrLatinE

13812 \newcommand*{\glsxtrLatinE}{%
13813 e\string=\glshex 2091,E
13814 }

\glsxtrLatinH

13815 \newcommand*{\glsxtrLatinH}{%
13816 h\string=\glshex 2095,H
13817 }

\glsxtrLatinI

13818 \newcommand*{\glsxtrLatinI}{%
13819 i\string=\glshex 2071,I
13820 }

\glsxtrLatinK

13821 \newcommand*{\glsxtrLatinK}{%
13822 k\string=\glshex 2096,K
13823 }

\glsxtrLatinL

13824 \newcommand*{\glsxtrLatinL}{%
13825 l\string=\glshex 2097,L
13826 }

\glsxtrLatinM

13827 \newcommand*{\glsxtrLatinM}{%
13828 m\string=\glshex 2098,M
13829 }

\glsxtrLatinN

13830 \newcommand*{\glsxtrLatinN}{%
13831 n\string=\glshex 207F\string=\glshex 2099,N
13832 }

\glsxtrLatinO

13833 \newcommand*{\glsxtrLatinO}{%
13834 o\string=\glshex 00BA\string=\glshex 2092,O
13835 }

```

\glxtrLatinP
13836 \newcommand*{\glxtrLatinP}{%
13837   p\string=\glshex 209A,P
13838 }

\glxtrLatinS
13839 \newcommand*{\glxtrLatinS}{%
13840   s\string=\glshex 209B,S
13841 }

\glxtrLatinT
13842 \newcommand*{\glxtrLatinT}{%
13843   t\string=\glshex 209C,T
13844 }

\glxtrLatinX
13845 \newcommand*{\glxtrLatinX}{%
13846   x\string=\glshex 2093,X
13847 }

lsxtrLatinSchwa  Latin schwa (lower case, subscript and upper case).
13848 \newcommand*{\glxtrLatinSchwa}{%
13849   \glshex 0259\string=\glshex 2094,\glshex 018F
13850 }

trLatinEszettSs
13851 \newcommand*{\glxtrLatinEszettSs}{%
13852   \glshex 00DF% eszett
13853   \string=\glshex 017Fs % long S s
13854 }

trLatinEszettSz
13855 \newcommand*{\glxtrLatinEszettSz}{%
13856   \glshex 00DF% eszett
13857   \string= \glshex 017Fz % long S z
13858 }

\glxtrLatinEth
13859 \newcommand*{\glxtrLatinEth}{%
13860   \glshex 00F0,\glshex 00D0% eth
13861 }

lsxtrLatinThorn
13862 \newcommand*{\glxtrLatinThorn}{%
13863   \glshex 00FE,\glshex 00DE% thorn
13864 }

```

LatinAELigature

```
13865 \newcommand*{\glxtrLatinAELigature}{%
13866   \glshex 00E6,\glshex 00C6% AE-ligature
13867 }
```

LatinOELigature

```
13868 \newcommand*{\glxtrLatinOELigature}{%
13869   \glshex 0153,\glshex 0152% OE-ligature
13870 }
```

\glxtrLatinAA

```
13871 \newcommand*{\glxtrLatinAA}{%
13872   \glshex 00E5=a\glshex 030A,% \aa
13873   \glshex 00C5=A\glshex 030A% \AA
13874 }
```

glxtrLatinWynn

```
13875 \newcommand*{\glxtrLatinWynn}{%
13876   \glshex 01BF,\glshex 01F7% wynn
13877 }
```

trLatinInsularG

```
13878 \newcommand*{\glxtrLatinInsularG}{%
13879   \glshex 1D79,\glshex A77D% insular G
13880   \string; g, G
13881 }
```

sxtrLatinOslash

```
13882 \newcommand*{\glxtrLatinOslash}{%
13883   \glshex 00F8,\glshex 00D8% \o, \O
13884 }
```

sxtrLatinLslash

```
13885 \newcommand*{\glxtrLatinLslash}{%
13886   \glshex 0142,\glshex 0141% \l, \L
13887 }
```

thUpGreekIrules Includes digamma between epsilon and zeta.

```
13888 \newcommand*{\glxtrMathUpGreekIrules}{%
13889   \glxtrUpAlpha
13890   \string<\glxtrUpBeta
13891   \string<\glxtrUpGamma
13892   \string<\glxtrUpDelta
13893   \string<\glxtrUpEpsilon
13894   \string<\glxtrUpDigamma
13895   \string<\glxtrUpZeta
13896   \string<\glxtrUpEta
13897   \string<\glxtrUpTheta
```

```

13898 \string<\glxtrUpIota
13899 \string<\glxtrUpKappa
13900 \string<\glxtrUpLambda
13901 \string<\glxtrUpMu
13902 \string<\glxtrUpNu
13903 \string<\glxtrUpXi
13904 \string<\glxtrUpOmicron
13905 \string<\glxtrUpPi
13906 \string<\glxtrUpRho
13907 \string<\glxtrUpSigma
13908 \string<\glxtrUpTau
13909 \string<\glxtrUpUpsilon
13910 \string<\glxtrUpPhi
13911 \string<\glxtrUpChi
13912 \string<\glxtrUpPsi
13913 \string<\glxtrUpOmega
13914 }

```

`\hUpGreekIIrules` Doesn't include digamma.

```

13915 \newcommand*{\glxtrMathUpGreekIIrules}{%
13916 \glxtrUpAlpha
13917 \string<\glxtrUpBeta
13918 \string<\glxtrUpGamma
13919 \string<\glxtrUpDelta
13920 \string<\glxtrUpEpsilon
13921 \string<\glxtrUpZeta
13922 \string<\glxtrUpEta
13923 \string<\glxtrUpTheta
13924 \string<\glxtrUpIota
13925 \string<\glxtrUpKappa
13926 \string<\glxtrUpLambda
13927 \string<\glxtrUpMu
13928 \string<\glxtrUpNu
13929 \string<\glxtrUpXi
13930 \string<\glxtrUpOmicron
13931 \string<\glxtrUpPi
13932 \string<\glxtrUpRho
13933 \string<\glxtrUpSigma
13934 \string<\glxtrUpTau
13935 \string<\glxtrUpUpsilon
13936 \string<\glxtrUpPhi
13937 \string<\glxtrUpChi
13938 \string<\glxtrUpPsi
13939 \string<\glxtrUpOmega
13940 }

```

`\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.

```

13941 \newcommand*{\glxtrMathItalicGreekIrules}{%

```



```

13942 \glxtrMathItalicAlpha
13943 \string<\glxtrMathItalicBeta
13944 \string<\glxtrMathItalicGamma
13945 \string<\glxtrMathItalicDelta
13946 \string<\glxtrMathItalicEpsilon
13947 \string<\glxtrUpDigamma
13948 \string<\glxtrMathItalicZeta
13949 \string<\glxtrMathItalicEta
13950 \string<\glxtrMathItalicTheta
13951 \string<\glxtrMathItalicIota
13952 \string<\glxtrMathItalicKappa
13953 \string<\glxtrMathItalicLambda
13954 \string<\glxtrMathItalicMu
13955 \string<\glxtrMathItalicNu
13956 \string<\glxtrMathItalicXi
13957 \string<\glxtrMathItalicOmicron
13958 \string<\glxtrMathItalicPi
13959 \string<\glxtrMathItalicRho
13960 \string<\glxtrMathItalicSigma
13961 \string<\glxtrMathItalicTau
13962 \string<\glxtrMathItalicUpsilon
13963 \string<\glxtrMathItalicPhi
13964 \string<\glxtrMathItalicChi
13965 \string<\glxtrMathItalicPsi
13966 \string<\glxtrMathItalicOmega
13967 }

```

licGreekIIrules Doesn't include digamma.

```

13968 \newcommand*{\glxtrMathItalicGreekIIrules}{%
13969 \glxtrMathItalicAlpha
13970 \string<\glxtrMathItalicBeta
13971 \string<\glxtrMathItalicGamma
13972 \string<\glxtrMathItalicDelta
13973 \string<\glxtrMathItalicEpsilon
13974 \string<\glxtrMathItalicZeta
13975 \string<\glxtrMathItalicEta
13976 \string<\glxtrMathItalicTheta
13977 \string<\glxtrMathItalicIota
13978 \string<\glxtrMathItalicKappa
13979 \string<\glxtrMathItalicLambda
13980 \string<\glxtrMathItalicMu
13981 \string<\glxtrMathItalicNu
13982 \string<\glxtrMathItalicXi
13983 \string<\glxtrMathItalicOmicron
13984 \string<\glxtrMathItalicPi
13985 \string<\glxtrMathItalicRho
13986 \string<\glxtrMathItalicSigma
13987 \string<\glxtrMathItalicTau
13988 \string<\glxtrMathItalicUpsilon

```

```

13989 \string<\glxtrMathItalicPhi
13990 \string<\glxtrMathItalicChi
13991 \string<\glxtrMathItalicPsi
13992 \string<\glxtrMathItalicOmega
13993 }

```

UpperGreekIrules Upper case only (includes upright digamma).

```

13994 \newcommand*{\glxtrMathItalicUpperGreekIrules}{%
13995 \glshex 1D6E2% upper case alpha (maths italic)
13996 \string<\glshex 1D6E3% upper case beta (maths italic)
13997 \string<\glshex 1D6E4% upper case gamma (maths italic)
13998 \string<\glshex 1D6E5% upper case delta (maths italic)
13999 \string<\glshex 1D6E6% upper case epsilon (maths italic)
14000 \string<\glshex 03DC% upper case digamma
14001 \string<\glshex 1D6E7% upper case zeta (maths italic)
14002 \string<\glshex 1D6E8% upper case eta (maths italic)
14003 \string<\glshex 1D6E9% upper case theta (maths italic)
14004 \string<\glshex 1D6F3% upper case theta variant (maths italic)
14005 \string<\glshex 1D6EA% upper case iota (maths italic)
14006 \string<\glshex 1D6EB% upper case kappa (maths italic)
14007 \string<\glshex 1D6EC% upper case lambda (maths italic)
14008 \string<\glshex 1D6ED% upper case mu (maths italic)
14009 \string<\glshex 1D6EE% upper case nu (maths italic)
14010 \string<\glshex 1D6EF% upper case xi (maths italic)
14011 \string<\glshex 1D6F0% upper case omicron (maths italic)
14012 \string<\glshex 1D6F1% upper case pi (maths italic)
14013 \string<\glshex 1D6F2% upper case rho (maths italic)
14014 \string<\glshex 1D6F4% upper case sigma (maths italic)
14015 \string<\glshex 1D6F5% upper case tau (maths italic)
14016 \string<\glshex 1D6F6% upper case upsilon (maths italic)
14017 \string<\glshex 1D6F7% upper case phi (maths italic)
14018 \string<\glshex 1D6F8% upper case chi (maths italic)
14019 \string<\glshex 1D6F9% upper case psi (maths italic)
14020 \string<\glshex 1D6FA% upper case omega (maths italic)
14021 }

```

UpperGreekIIrules Upper case only (doesn't include upright digamma).

```

14022 \newcommand*{\glxtrMathItalicUpperGreekIIrules}{%
14023 \glshex 1D6E2% upper case alpha (maths italic)
14024 \string<\glshex 1D6E3% upper case beta (maths italic)
14025 \string<\glshex 1D6E4% upper case gamma (maths italic)
14026 \string<\glshex 1D6E5% upper case delta (maths italic)
14027 \string<\glshex 1D6E6% upper case epsilon (maths italic)
14028 \string<\glshex 1D6E7% upper case zeta (maths italic)
14029 \string<\glshex 1D6E8% upper case eta (maths italic)
14030 \string<\glshex 1D6E9% upper case theta (maths italic)
14031 \string<\glshex 1D6F3% upper case theta variant (maths italic)
14032 \string<\glshex 1D6EA% upper case iota (maths italic)
14033 \string<\glshex 1D6EB% upper case kappa (maths italic)

```

```

14034 \string<\glshex 1D6EC% upper case lambda (maths italic)
14035 \string<\glshex 1D6ED% upper case mu (maths italic)
14036 \string<\glshex 1D6EE% upper case nu (maths italic)
14037 \string<\glshex 1D6EF% upper case xi (maths italic)
14038 \string<\glshex 1D6F0% upper case omicron (maths italic)
14039 \string<\glshex 1D6F1% upper case pi (maths italic)
14040 \string<\glshex 1D6F2% upper case rho (maths italic)
14041 \string<\glshex 1D6F4% upper case sigma (maths italic)
14042 \string<\glshex 1D6F5% upper case tau (maths italic)
14043 \string<\glshex 1D6F6% upper case upsilon (maths italic)
14044 \string<\glshex 1D6F7% upper case phi (maths italic)
14045 \string<\glshex 1D6F8% upper case chi (maths italic)
14046 \string<\glshex 1D6F9% upper case psi (maths italic)
14047 \string<\glshex 1D6FA% upper case omega (maths italic)
14048 }

```

LowerGreekIrules Lower case only (includes upright digamma).

```

14049 \newcommand*{\glstrMathItalicLowerGreekIrules}{%
14050 \glshex 1D6FC% lower case alpha (maths italic)
14051 \string<\glshex 1D6FD% lower case beta (maths italic)
14052 \string<\glshex 1D6FE% lower case gamma (maths italic)
14053 \string<\glshex 1D6FF% lower case delta (maths italic)
14054 \string<\glshex 1D700% lower case epsilon (maths italic)
14055 \string=<\glshex 1D716% lower case epsilon variant (maths italic)
14056 \string<\glshex 03DD% lower case digamma
14057 \string<\glshex 1D701% lower case zeta (maths italic)
14058 \string<\glshex 1D702% lower case eta (maths italic)
14059 \string<\glshex 1D703% lower case theta (maths italic)
14060 \string=<\glshex 1D717% lower case theta variant (maths italic)
14061 \string<\glshex 1D704% lower case iota (maths italic)
14062 \string<\glshex 1D705% lower case kappa (maths italic)
14063 \string=<\glshex 1D718% lower case kappa variant (maths italic)
14064 \string<\glshex 1D706% lower case lambda (maths italic)
14065 \string<\glshex 1D707% lower case mu (maths italic)
14066 \string<\glshex 1D708% lower case nu (maths italic)
14067 \string<\glshex 1D709% lower case xi (maths italic)
14068 \string<\glshex 1D70A% lower case omicron (maths italic)
14069 \string<\glshex 1D70B% lower case pi (maths italic)
14070 \string=<\glshex 1D71B% lower case pi variant (maths italic)
14071 \string<\glshex 1D70C% lower case rho (maths italic)
14072 \string=<\glshex 1D71A% lower case rho variant (maths italic)
14073 \string<\glshex 1D70D% lower case final sigma (maths italic)
14074 \string=<\glshex 1D70E% lower case sigma (maths italic)
14075 \string<\glshex 1D70F% lower case tau (maths italic)
14076 \string<\glshex 1D710% lower case upsilon (maths italic)
14077 \string<\glshex 1D711% lower case phi (maths italic)
14078 \string=<\glshex 1D719% lower case phi variant (maths italic)
14079 \string<\glshex 1D712% lower case chi (maths italic)
14080 \string<\glshex 1D713% lower case psi (maths italic)

```

```

14081 \string<\glshex 1D714% lower case omega (maths italic)
14082 }

```

LowerGreeklIrules Lower case only (doesn't includes upright digamma).

```

14083 \newcommand*{\glxtrMathItalicLowerGreeklIrules}{%
14084 \glshex 1D6FC% lower case alpha (maths italic)
14085 \string<\glshex 1D6FD% lower case beta (maths italic)
14086 \string<\glshex 1D6FE% lower case gamma (maths italic)
14087 \string<\glshex 1D6FF% lower case delta (maths italic)
14088 \string<\glshex 1D700% lower case epsilon (maths italic)
14089 \string=>\glshex 1D716% lower case epsilon variant (maths italic)
14090 \string<\glshex 1D701% lower case zeta (maths italic)
14091 \string<\glshex 1D702% lower case eta (maths italic)
14092 \string<\glshex 1D703% lower case theta (maths italic)
14093 \string=>\glshex 1D717% lower case theta variant (maths italic)
14094 \string<\glshex 1D704% lower case iota (maths italic)
14095 \string<\glshex 1D705% lower case kappa (maths italic)
14096 \string=>\glshex 1D718% lower case kappa variant (maths italic)
14097 \string<\glshex 1D706% lower case lambda (maths italic)
14098 \string<\glshex 1D707% lower case mu (maths italic)
14099 \string<\glshex 1D708% lower case nu (maths italic)
14100 \string<\glshex 1D709% lower case xi (maths italic)
14101 \string<\glshex 1D70A% lower case omicron (maths italic)
14102 \string<\glshex 1D70B% lower case pi (maths italic)
14103 \string=>\glshex 1D71B% lower case pi variant (maths italic)
14104 \string<\glshex 1D70C% lower case rho (maths italic)
14105 \string=>\glshex 1D71A% lower case rho variant (maths italic)
14106 \string<\glshex 1D70D% lower case final sigma (maths italic)
14107 \string=>\glshex 1D70E% lower case sigma (maths italic)
14108 \string<\glshex 1D70F% lower case tau (maths italic)
14109 \string<\glshex 1D710% lower case upsilon (maths italic)
14110 \string<\glshex 1D711% lower case phi (maths italic)
14111 \string=>\glshex 1D719% lower case phi variant (maths italic)
14112 \string<\glshex 1D712% lower case chi (maths italic)
14113 \string<\glshex 1D713% lower case psi (maths italic)
14114 \string<\glshex 1D714% lower case omega (maths italic)
14115 }

```

MathGreeklIrules Includes both upright and italic with digamma between epsilon and zeta.

```

14116 \newcommand*{\glxtrMathGreeklIrules}{%
14117 \glxtrMathItalicAlpha
14118 \string;\glxtrUpAlpha
14119 \string<\glxtrMathItalicBeta
14120 \string;\glxtrUpBeta
14121 \string<\glxtrMathItalicGamma
14122 \string;\glxtrUpGamma
14123 \string<\glxtrMathItalicDelta
14124 \string;\glxtrUpDelta
14125 \string<\glxtrMathItalicEpsilon

```

```

14126 \string;\glxtrUpEpsilon
14127 \string<\glxtrUpDigamma
14128 \string<\glxtrMathItalicZeta
14129 \string;\glxtrUpZeta
14130 \string<\glxtrMathItalicEta
14131 \string;\glxtrUpEta
14132 \string<\glxtrMathItalicTheta
14133 \string;\glxtrUpTheta
14134 \string<\glxtrMathItalicIota
14135 \string;\glxtrUpIota
14136 \string<\glxtrMathItalicKappa
14137 \string;\glxtrUpKappa
14138 \string<\glxtrMathItalicLambda
14139 \string;\glxtrUpLambda
14140 \string<\glxtrMathItalicMu
14141 \string;\glxtrUpMu
14142 \string<\glxtrMathItalicNu
14143 \string;\glxtrUpNu
14144 \string<\glxtrMathItalicXi
14145 \string;\glxtrUpXi
14146 \string<\glxtrMathItalicOmicron
14147 \string;\glxtrUpOmicron
14148 \string<\glxtrMathItalicPi
14149 \string;\glxtrUpPi
14150 \string<\glxtrMathItalicRho
14151 \string;\glxtrUpRho
14152 \string<\glxtrMathItalicSigma
14153 \string;\glxtrUpSigma
14154 \string<\glxtrMathItalicTau
14155 \string;\glxtrUpTau
14156 \string<\glxtrMathItalicUpsilon
14157 \string;\glxtrUpUpsilon
14158 \string<\glxtrMathItalicPhi
14159 \string;\glxtrUpPhi
14160 \string<\glxtrMathItalicChi
14161 \string;\glxtrUpChi
14162 \string<\glxtrMathItalicPsi
14163 \string;\glxtrUpPsi
14164 \string<\glxtrMathItalicOmega
14165 \string;\glxtrUpOmega
14166 }

```

mathGreekIIrules Includes both upright and italic (digamma not included).

```

14167 \newcommand*{\glxtrMathGreekIIrules}{%
14168 \glxtrMathItalicAlpha
14169 \string;\glxtrUpAlpha
14170 \string<\glxtrMathItalicBeta
14171 \string;\glxtrUpBeta
14172 \string<\glxtrMathItalicGamma

```

```

14173 \string;\glsxtrUpGamma
14174 \string<\glsxtrMathItalicDelta
14175 \string;\glsxtrUpDelta
14176 \string<\glsxtrMathItalicEpsilon
14177 \string;\glsxtrUpEpsilon
14178 \string<\glsxtrMathItalicZeta
14179 \string;\glsxtrUpZeta
14180 \string<\glsxtrMathItalicEta
14181 \string;\glsxtrUpEta
14182 \string<\glsxtrMathItalicTheta
14183 \string;\glsxtrUpTheta
14184 \string<\glsxtrMathItalicIota
14185 \string;\glsxtrUpIota
14186 \string<\glsxtrMathItalicKappa
14187 \string;\glsxtrUpKappa
14188 \string<\glsxtrMathItalicLambda
14189 \string;\glsxtrUpLambda
14190 \string<\glsxtrMathItalicMu
14191 \string;\glsxtrUpMu
14192 \string<\glsxtrMathItalicNu
14193 \string;\glsxtrUpNu
14194 \string<\glsxtrMathItalicXi
14195 \string;\glsxtrUpXi
14196 \string<\glsxtrMathItalicOmicron
14197 \string;\glsxtrUpOmicron
14198 \string<\glsxtrMathItalicPi
14199 \string;\glsxtrUpPi
14200 \string<\glsxtrMathItalicRho
14201 \string;\glsxtrUpRho
14202 \string<\glsxtrMathItalicSigma
14203 \string;\glsxtrUpSigma
14204 \string<\glsxtrMathItalicTau
14205 \string;\glsxtrUpTau
14206 \string<\glsxtrMathItalicUpsilon
14207 \string;\glsxtrUpUpsilon
14208 \string<\glsxtrMathItalicPhi
14209 \string;\glsxtrUpPhi
14210 \string<\glsxtrMathItalicChi
14211 \string;\glsxtrUpChi
14212 \string<\glsxtrMathItalicPsi
14213 \string;\glsxtrUpPsi
14214 \string<\glsxtrMathItalicOmega
14215 \string;\glsxtrUpOmega
14216 }

```

`\glsxtrUpAlpha`

```

14217 \newcommand*{\glsxtrUpAlpha}{%
14218 \glshex 03B1,% lower case alpha
14219 \glshex 0391% upper case alpha

```

14220 }

\glsxtrUpBeta

```
14221 \newcommand*{\glsxtrUpBeta}{%
14222   \glshex 03B2,% lower case beta
14223   \glshex 0392% upper case beta
14224 }
```

\glsxtrUpGamma

```
14225 \newcommand*{\glsxtrUpGamma}{%
14226   \glshex 03B3,% lower case gamma
14227   \glshex 0393% upper case gamma
14228 }
```

\glsxtrUpDelta

```
14229 \newcommand*{\glsxtrUpDelta}{%
14230   \glshex 03B4,% lower case delta
14231   \glshex 0394% upper case delta
14232 }
```

glsxtrUpEpsilon

```
14233 \newcommand*{\glsxtrUpEpsilon}{%
14234   \glshex 03B5% lower case epsilon
14235   \string=\glshex 03F5,% lower case epsilon variant
14236   \glshex 0395% upper case epsilon
14237 }
```

glsxtrUpDigamma

```
14238 \newcommand*{\glsxtrUpDigamma}{%
14239   \glshex 03DD,% lower case digamma
14240   \glshex 03DC% upper case digamma
14241 }
```

\glsxtrUpZeta

```
14242 \newcommand*{\glsxtrUpZeta}{%
14243   \glshex 03B6,% lower case zeta
14244   \glshex 0396% upper case zeta
14245 }
```

\glsxtrUpEta

```
14246 \newcommand*{\glsxtrUpEta}{%
14247   \glshex 03B7,% lower case eta
14248   \glshex 0397% upper case eta
14249 }
```

\glsxtrUpTheta

```
14250 \newcommand*{\glsxtrUpTheta}{%
14251   \glshex 03B8% lower case theta
```

```

14252 \string=\glshex 03D1,% lower case theta variant
14253 \glshex 0398% upper case theta
14254 }

```

$\backslash\mathrm{glxtrUpIota}$

```

14255 \newcommand*{\glxtrUpIota}{%
14256 \glshex 03B9,% lower case iota
14257 \glshex 0399% upper case iota
14258 }

```

$\backslash\mathrm{glxtrUpKappa}$

```

14259 \newcommand*{\glxtrUpKappa}{%
14260 \glshex 03BA,% lower case kappa
14261 \string=\glshex 03F0,% lower case kappa variant
14262 \glshex 039A% upper case kappa
14263 }

```

$\backslash\mathrm{glxtrUpLambda}$

```

14264 \newcommand*{\glxtrUpLambda}{%
14265 \glshex 03BB,% lower lambda
14266 \glshex 039B% upper case lambda
14267 }

```

$\backslash\mathrm{glxtrUpMu}$

```

14268 \newcommand*{\glxtrUpMu}{%
14269 \glshex 03BC,% lower case mu
14270 \glshex 039C% upper case mu
14271 }

```

$\backslash\mathrm{glxtrUpNu}$

```

14272 \newcommand*{\glxtrUpNu}{%
14273 \glshex 03BD,% lower case nu
14274 \glshex 039D% upper case nu
14275 }

```

$\backslash\mathrm{glxtrUpXi}$

```

14276 \newcommand*{\glxtrUpXi}{%
14277 \glshex 03BE,% lower case xi
14278 \glshex 039E% upper case xi
14279 }

```

$\mathrm{glxtrUpOmicron}$

```

14280 \newcommand*{\glxtrUpOmicron}{%
14281 \glshex 03BF,% lower case omicron
14282 \glshex 039F% upper case omicron
14283 }

```


\glxtrUpPi

```
14284 \newcommand*{\glxtrUpPi}{%
14285   \glshex 03C0% lower case pi
14286   \string=\glshex 03D6,% lower case pi variant
14287   \glshex 03A0% upper case pi
14288 }
```

\glxtrUpRho

```
14289 \newcommand*{\glxtrUpRho}{%
14290   \glshex 03C1% lower case rho
14291   \string=\glshex 03F1,% lower case rho variant
14292   \glshex 03A1% upper case rho
14293 }
```

\glxtrUpSigma

```
14294 \newcommand*{\glxtrUpSigma}{%
14295   \glshex 03C2% lower case sigma
14296   \string=\glshex 03C3,% lower case sigma
14297   \glshex 03A3% upper case sigma
14298 }
```

\glxtrUpTau

```
14299 \newcommand*{\glxtrUpTau}{%
14300   \glshex 03C4,% lower case tau
14301   \glshex 03A4% upper case tau
14302 }
```

\glxtrUpUpsilon

```
14303 \newcommand*{\glxtrUpUpsilon}{%
14304   \glshex 03C5,% lower case upsilon
14305   \glshex 03A5% upper case upsilon
14306 }
```

\glxtrUpPhi

```
14307 \newcommand*{\glxtrUpPhi}{%
14308   \glshex 03C6% lower case phi
14309   \string=\glshex 03D5,% lower case phi variant
14310   \glshex 03A6% upper case phi
14311 }
```

\glxtrUpChi

```
14312 \newcommand*{\glxtrUpChi}{%
14313   \glshex 03C7,% lower case chi
14314   \glshex 03A7% upper case chi
14315 }
```

\glxtrUpPsi

```
14316 \newcommand*{\glxtrUpPsi}{%
```

```

14317 \glshex 03C8,% lower case psi
14318 \glshex 03A8% upper case psi
14319 }

```

$\backslash\mathrm{glxtrUpOmega}$

```

14320 \newcommand*\glxtrUpOmega{%
14321 \glshex 03C9,% lower case omega
14322 \glshex 03A9% upper case omega
14323 }

```

$\mathrm{MathItalicAlpha}$

```

14324 \newcommand*\glxtrMathItalicAlpha{%
14325 \glshex 1D6FC,% lower case alpha (maths italic)
14326 \glshex 1D6E2% upper case alpha (maths italic)
14327 }

```

$\mathrm{rMathItalicBeta}$

```

14328 \newcommand*\glxtrMathItalicBeta{%
14329 \glshex 1D6FD,% lower case beta (maths italic)
14330 \glshex 1D6E3% upper case beta (maths italic)
14331 }

```

$\mathrm{MathItalicGamma}$

```

14332 \newcommand*\glxtrMathItalicGamma{%
14333 \glshex 1D6FE,% lower case gamma (maths italic)
14334 \glshex 1D6E4% upper case gamma (maths italic)
14335 }

```

$\mathrm{MathItalicDelta}$

```

14336 \newcommand*\glxtrMathItalicDelta{%
14337 \glshex 1D6FF,% lower case delta (maths italic)
14338 \glshex 1D6E5% upper case delta (maths italic)
14339 }

```

$\mathrm{thItalicEpsilon}$

```

14340 \newcommand*\glxtrMathItalicEpsilon{%
14341 \glshex 1D700% lower case epsilon (maths italic)
14342 \string=\glshex 1D716,% lower case epsilon variant (maths italic)
14343 \glshex 1D6E6% upper case epsilon (maths italic)
14344 }

```

$\mathrm{rMathItalicZeta}$

```

14345 \newcommand*\glxtrMathItalicZeta{%
14346 \glshex 1D701,% lower case zeta (maths italic)
14347 \glshex 1D6E7% upper case zeta (maths italic)
14348 }

```

trMathItalicEta

```
14349 \newcommand*{\glxtrMathItalicEta}{%
14350   \glshex 1D702,% lower case eta (maths italic)
14351   \glshex 1D6E8% upper case eta (maths italic)
14352 }
```

MathItalicTheta

```
14353 \newcommand*{\glxtrMathItalicTheta}{%
14354   \glshex 1D703% lower case theta (maths italic)
14355   \string=\glshex 1D717,% lower case theta variant (maths italic)
14356   \glshex 1D6E9% upper case theta (maths italic)
14357   \string=\glshex 1D6F3% upper case theta variant (maths italic)
14358 }
```

rMathItalicIota

```
14359 \newcommand*{\glxtrMathItalicIota}{%
14360   \glshex 1D704,% lower case iota (maths italic)
14361   \glshex 1D6EA% upper case iota (maths italic)
14362 }
```

MathItalicKappa

```
14363 \newcommand*{\glxtrMathItalicKappa}{%
14364   \glshex 1D705% lower case kappa (maths italic)
14365   \string=\glshex 1D718,% lower case kappa variant (maths italic)
14366   \glshex 1D6EB% upper case kappa (maths italic)
14367 }
```

athItalicLambda

```
14368 \newcommand*{\glxtrMathItalicLambda}{%
14369   \glshex 1D706,% lower case lambda (maths italic)
14370   \glshex 1D6EC% upper case lambda (maths italic)
14371 }
```

xtrMathItalicMu

```
14372 \newcommand*{\glxtrMathItalicMu}{%
14373   \glshex 1D707,% lower case mu (maths italic)
14374   \glshex 1D6ED% upper case mu (maths italic)
14375 }
```

xtrMathItalicNu

```
14376 \newcommand*{\glxtrMathItalicNu}{%
14377   \glshex 1D708,% lower case nu (maths italic)
14378   \glshex 1D6EE% upper case nu (maths italic)
14379 }
```

xtrMathItalicXi

```
14380 \newcommand*{\glxtrMathItalicXi}{%
14381   \glshex 1D709,% lower case xi (maths italic)
```

```

14382 \glshex 1D6EF% upper case xi (maths italic)
14383 }

```

thItalicOmicron

```

14384 \newcommand*{\glxtrMathItalicOmicron}{%
14385 \glshex 1D70A,% lower case omicron (maths italic)
14386 \glshex 1D6F0% upper case omicron (maths italic)
14387 }

```

xtrMathItalicPi

```

14388 \newcommand*{\glxtrMathItalicPi}{%
14389 \glshex 1D70B% lower case pi (maths italic)
14390 \string=\glshex 1D71B,% lower case pi variant (maths italic)
14391 \glshex 1D6F1% upper case pi (maths italic)
14392 }

```

trMathItalicRho

```

14393 \newcommand*{\glxtrMathItalicRho}{%
14394 \glshex 1D70C% lower case rho (maths italic)
14395 \string=\glshex 1D71A,% lower case rho variant (maths italic)
14396 \glshex 1D6F2% upper case rho (maths italic)
14397 }

```

MathItalicSigma

```

14398 \newcommand*{\glxtrMathItalicSigma}{%
14399 \glshex 1D70D% lower case final sigma (maths italic)
14400 \string=\glshex 1D70E,% lower case sigma (maths italic)
14401 \glshex 1D6F4% upper case sigma (maths italic)
14402 }

```

trMathItalicTau

```

14403 \newcommand*{\glxtrMathItalicTau}{%
14404 \glshex 1D70F,% lower case tau (maths italic)
14405 \glshex 1D6F5% upper case tau (maths italic)
14406 }

```

thItalicUpsilon

```

14407 \newcommand*{\glxtrMathItalicUpsilon}{%
14408 \glshex 1D710,% lower case upsilon (maths italic)
14409 \glshex 1D6F6% upper case upsilon (maths italic)
14410 }

```

trMathItalicPhi

```

14411 \newcommand*{\glxtrMathItalicPhi}{%
14412 \glshex 1D711% lower case phi (maths italic)
14413 \string=\glshex 1D719,% lower case phi variant (maths italic)
14414 \glshex 1D6F7% upper case phi (maths italic)
14415 }

```

trMathItalicChi

```
14416 \newcommand*{\glxtrMathItalicChi}{%
14417   \glshex 1D712,% lower case chi (maths italic)
14418   \glshex 1D6F8% upper case chi (maths italic)
14419 }
```

trMathItalicPsi

```
14420 \newcommand*{\glxtrMathItalicPsi}{%
14421   \glshex 1D713,% lower case psi (maths italic)
14422   \glshex 1D6F9% upper case psi (maths italic)
14423 }
```

MathItalicOmega

```
14424 \newcommand*{\glxtrMathItalicOmega}{%
14425   \glshex 1D714,% lower case omega (maths italic)
14426   \glshex 1D6FA% upper case omega (maths italic)
14427 }
```

thItalicPartial

```
14428 \newcommand*{\glxtrMathItalicPartial}{%
14429   \glshex 1D715% partial differential (maths italic)
14430 }
```

MathItalicNabla

```
14431 \newcommand*{\glxtrMathItalicNabla}{%
14432   \glshex 1D6FB% nabla (maths italic)
14433 }
```

lsxtrdigitrules Digits from the Basic Latin set and subscript and superscript digit rules.

```
14434 \newcommand*{\glxtrdigitrules}{%
14435   0\string=\glshex 2080\string=\glshex 2070
14436   \string<1\string=\glshex 2081\string=\glshex 00B9
14437   \string<2\string=\glshex 2082\string=\glshex 00B2
14438   \string<3\string=\glshex 2083\string=\glshex 00B3
14439   \string<4\string=\glshex 2084\string=\glshex 2074
14440   \string<5\string=\glshex 2085\string=\glshex 2075
14441   \string<6\string=\glshex 2086\string=\glshex 2076
14442   \string<7\string=\glshex 2087\string=\glshex 2077
14443   \string<8\string=\glshex 2088\string=\glshex 2078
14444   \string<9\string=\glshex 2089\string=\glshex 2079
14445 }
```

BasicDigitrules Digits from the Basic Latin set.

```
14446 \newcommand*{\glxtrBasicDigitrules}{%
14447   0\string<1\string<2\string<3\string<4%
14448   \string<5\string<6\string<7\string<8\string<9%
14449 }
```

criptDigitrules Subscript digits.

```
14450 \newcommand*{\glxtrSubScriptDigitrules}{%
14451 \glshex 2080% subscript 0
14452 \string<\glshex 2081% subscript 1
14453 \string<\glshex 2082% subscript 2
14454 \string<\glshex 2083% subscript 3
14455 \string<\glshex 2084% subscript 4
14456 \string<\glshex 2085% subscript 5
14457 \string<\glshex 2086% subscript 6
14458 \string<\glshex 2087% subscript 7
14459 \string<\glshex 2088% subscript 8
14460 \string<\glshex 2089% subscript 9
14461 }
```

criptDigitrules Superscript digits.

```
14462 \newcommand*{\glxtrSuperScriptDigitrules}{%
14463 \glshex 2070% superscript 0
14464 \string<\glshex 00B9% superscript 1
14465 \string<\glshex 00B2% superscript 2
14466 \string<\glshex 00B3% superscript 3
14467 \string<\glshex 2074% superscript 4
14468 \string<\glshex 2075% superscript 5
14469 \string<\glshex 2076% superscript 6
14470 \string<\glshex 2077% superscript 7
14471 \string<\glshex 2078% superscript 8
14472 \string<\glshex 2079% superscript 9
14473 }
```

trfractionrules Vulgar fractions.

```
14474 \newcommand*{\glxtrfractionrules}{%
14475 \glshex 215F% fraction numerator one (1/)
14476 \string<\glshex 2189% zero thirds (0/3 = 0)
14477 \string<\glshex 2152% one tenth (1/10 = 0.1)
14478 \string<\glshex 2151% one ninth (1/9 ~ 0.111)
14479 \string<\glshex 215B% one eighth (1/8 = 0.125)
14480 \string<\glshex 2150% one seventh (1/7 ~ 0.143)
14481 \string<\glshex 2159% one sixth (1/6 ~ 0.167)
14482 \string<\glshex 2155% one fifth (1/5 = 0.2)
14483 \string<\glshex 00BC% one quarter (1/4 = 0.25)
14484 \string<\glshex 2153% one third (1/3 ~ 0.333)
14485 \string<\glshex 215C% three eighths (3/8 = 0.375)
14486 \string<\glshex 2156% two fifths (2/5 = 0.4)
14487 \string<\glshex 00BD% one half (1/2 = 0.5)
14488 \string<\glshex 2157% three fifths (3/5 = 0.6)
14489 \string<\glshex 215D% five eighths (5/8 = 0.625)
14490 \string<\glshex 2154% two thirds (2/3 ~ 0.667)
14491 \string<\glshex 00BE% three quarters (3/4 = 0.75)
14492 \string<\glshex 2158% four fifths (4/5 = 0.8)
14493 \string<\glshex 215A% five sixths (5/6 ~ 0.833)
```

```

14494 \string<\glshex 215E% seven eighths (7/8 = 0.875)
14495 }

```

sxtrdialecthook Check for scripts associated with the document dialects.

```

14496 \renewcommand{\@glsxtrdialecthook}{%
14497   \ifundef\CurrentTrackedScript
14498   {%
14499     \TrackLangIfHasDefaultScript{\CurrentTrackedLanguage}%
14500     {%
14501       \edef\CurrentTrackedScript{%
14502         \TrackLangGetDefaultScript\CurrentTrackedLanguage}%
14503     }%
14504   }%
14505 }%
14506 {}%
14507 \ifdef\CurrentTrackedScript
14508 {%
14509   \let\gls@orgTrackLangRequireDialectPrefix\TrackLangRequireDialectPrefix
14510   \def\TrackLangRequireDialectPrefix{glossariesxtr-}%
14511   \let\CurrentTrackedTag\CurrentTrackedScript
14512   \IfFileExists{\TrackLangRequireDialectPrefix\CurrentTrackedTag.ldf}
14513   {\RequireGlossariesExtraLang{\CurrentTrackedTag}}%
14514   {}%
14515   \let\TrackLangRequireDialectPrefix\gls@orgTrackLangRequireDialectPrefix
14516 }%
14517 {}%
14518 }

```

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.

```

14519 \ifdef\glsxtr@loaddialect
14520 {%
14521   \@ifpackageloaded{tracklang}
14522   {%
14523     \AnyTrackedLanguages
14524     {%
14525       \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
14526     }%
14527   }%
14528 }
14529 {}
14530 }
14531 {}

```

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:

```
14532 \NeedsTeXFormat{LaTeX2e}
14533 \ProvidesPackage{glossaries-extra-stylemods}[2020/02/28 v1.43 (NLCT)]
```

Provide package options to automatically load required predefined styles. The simplest method is to just test for the existence of the file `glossary-<option>.sty`. Packages can't be loaded whilst the options are being processed, so save the list in `\@glxtr@loadstyles`.

`sxtr@loadstyles`

```
14534 \newcommand*{\@glxtr@loadstyles}{}
```

`all` Provide all known styles.

```
14535 \DeclareOption{all}{%
14536   \appto\@glxtr@loadstyles{%
14537     \RequirePackage{glossary-inline}%
14538     \RequirePackage{glossary-list}%
14539     \RequirePackage{glossary-tree}%
14540     \RequirePackage{glossary-mcols}%
14541     \RequirePackage{glossary-long}%
14542     \RequirePackage{glossary-longragged}%
14543     \RequirePackage{glossary-longbooktabs}%
14544     \RequirePackage{glossary-super}%
14545     \RequirePackage{glossary-superragged}%
14546     \RequirePackage{glossary-bookindex}%
14547     \RequirePackage{glossary-longextra}%
14548     \RequirePackage{glossary-topic}%
14549   }
14550 }

14551 \DeclareOption*{%
14552   \IfFileExists{glossary-\CurrentOption.sty}
14553   {\eappto\@glxtr@loadstyles{%
14554     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
14555   }%
```



```

14556   {%
14557     \PackageError{glossaries-extra-styles}%
14558     {Unknown option ‘\CurrentOption’}{}%
14559   }%
14560 }

```

Process the package options:

```
14561 \ProcessOptions
```

Load the required packages:

```
14562 \@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.

```
14563 \providecommand*{\glsxtrprelocation}{\space}
```

In case we have an old version of glossaries:

`ewglossarystyle`

```

14564 \providecommand{\renewglossarystyle}[2]{%
14565   \ifcsundef{@glsstyle@#1}%
14566   {%
14567     \PackageError{glossaries-extra}{Glossary style ‘#1’ isn’t already defined}{}%
14568   }%
14569   {%
14570     \csdef{@glsstyle@#1}{#2}%
14571   }%
14572 }

```

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.

```

14573 \ifdef{\@glsstyle@listdotted}
14574 {%
14575   \renewglossarystyle{listdotted}{%
14576     \setglossarystyle{list}%
14577     \renewcommand*{\glossentry}[2]{%
14578       \item[]\makebox[\glslistdottedwidth][l]{%
14579         \glsentryitem{##1}%
14580         \glstarget{##1}{\glossentryname{##1}}%
14581         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
14582         \glossentrydesc{##1}\glspostdescription}%
14583     \renewcommand*{\subglossentry}[3]{%
14584       \item[]\makebox[\glslistdottedwidth][l]{%
14585         \glssubentryitem{##2}%
14586         \glstarget{##2}{\glossentryname{##2}}%
14587         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%

```

```

14588 \glossentrydesc{##2}\glspostdescription}%
14589 }
14590 }
14591 {%

```

Assume the style isn't required if it hasn't already been defined.

```
14592 }
```

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.

```

14593 \ifdef{\@glsstyle@list}
14594 {%

```

listprelocation Space before number list for top-level entries.

```
14595 \newcommand{\glslistprelocation}{\glstrprelocation}
```

childprelocation Space before number list for child entries.

```
14596 \newcommand{\glslistchildprelocation}{\glslistprelocation}
```

childpostlocation Full stop after number list.

```
14597 \newcommand{\glslistchildpostlocation}{.}
```

\glslistdesc

```
14598 \newcommand{\glslistdesc}[1]{\glossentrydesc{#1}\glspostdescription}
```

lslistgroupskip

```
14599 \newcommand{\glslistgroupskip}{\nobreak\indexspace\nobreak}
```

Redefine list to use these commands.

```

14600 \renewglossarystyle{list}{%
14601 \renewenvironment{theglossary}%
14602 {\begin{description}}{\end{description}}%
14603 \renewcommand*\glossaryheader{}%
14604 \renewcommand*\glsgroupheading[1]{}%
14605 \renewcommand*\glossentry[2]{%
14606 \item[\glssentryitem{##1}]%
14607 \glstarget{##1}{\glossentryname{##1}}]
14608 \glslistdesc{##1}\glslistprelocation ##2}%
14609 \renewcommand*\subglossentry[3]{%
14610 \glssubentryitem{##2}%
14611 \glstarget{##2}{\strut}\space
14612 \glslistdesc{##2}%
14613 \glslistchildprelocation ##3\glslistchildpostlocation}%
14614 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\glslistgroupskip\fi}%
14615 }
14616 }
14617 {}

```

Similarly for altlist. Since it requires list, the new commands should have been defined above.

```

14618 \ifdef{\@glsstyle@altlist}
14619 {%
14620   \renewglossarystyle{altlist}{%
14621     \setglossarystyle{list}%
14622     \renewcommand*{\glossentry}[2]{%
14623       \item[\glsentryitem{##1}]%
14624       \glstarget{##1}{\glossentryname{##1}}}%
14625     \mbox{}\par\nobreak\@afterheading
14626     \glslistdesc{##1}\glslistprelocation ##2}%
14627   \renewcommand{\subglossentry}[3]{%
14628     \par
14629     \glssubentryitem{##2}%
14630     \glstarget{##2}{\strut}\glslistdesc{##2}%
14631     \glslistchildprelocation ##3}%
14632   }
14633 }
14634 {}

```

Redefine listgroup so that it discourages a break after group headings.

```

14635 \ifdef{\@glsstyle@listgroup}
14636 {%
14637   \renewglossarystyle{listgroup}{%
14638     \setglossarystyle{list}%
14639     \renewcommand*{\glsgroupheading}[1]{%
14640       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
14641       \mbox{}\par\nobreak\@afterheading
14642     }%
14643   }
14644 }
14645 {}

```

Similarly for listhypergroup.

```

14646 \ifdef{\@glsstyle@listhypergroup}
14647 {%
14648   \renewglossarystyle{listhypergroup}{%
14649     \setglossarystyle{list}%
14650     \renewcommand*{\glossaryheader}{%
14651       \glslistnavigationitem{\glsnavigation}}%
14652     \renewcommand*{\glsgroupheading}[1]{%
14653       \item[\glslistgroupheaderfmt
14654         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
14655       \mbox{}\par\nobreak\@afterheading
14656     }%
14657   }
14658 }
14659 {}

```

Similarly for altlistgroup.

```

14660 \ifdef{\@glsstyle@altlistgroup}

```

```

14661 {%
14662   \renewglossarystyle{altlistgroup}{%
14663     \setglossarystyle{altlist}%
14664     \renewcommand*{\glsgroupheading}[1]{%
14665       \item[\glslistgroupheaderfmt{\glsgetgrouptitle{##1}}]%
14666       \mbox{} \par \nobreak \afterheading
14667     }%
14668   }
14669 }
14670 {}

```

Similarly for altlisthypergroup.

```

14671 \ifdef{\@glsstyle@altlisthypergroup}
14672 {%
14673   \renewglossarystyle{altlisthypergroup}{%
14674     \setglossarystyle{altlist}%
14675     \renewcommand*{\glossaryheader}{%
14676       \glslistnavigationitem{\glsnavigation}}%
14677     \renewcommand*{\glsgroupheading}[1]{%
14678       \item[\glslistgroupheaderfmt
14679         {\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}]%
14680       \mbox{} \par \nobreak \afterheading
14681     }%
14682   }
14683 }
14684 {}

```

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`.

```

14685 \ifcsdef{@glsstyle@long}
14686 {%
14687   \renewglossarystyle{long}{%
14688     \renewenvironment{theglossary}%
14689       {\begin{longtable}{lp{\glsdescwidth}}}%
14690       {\end{longtable}}%
14691     \renewcommand*{\glossaryheader}{}%
14692     \renewcommand*{\glsgroupheading}[1]{}%
14693     \renewcommand{\glossentry}[2]{%
14694       \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14695       \glossentrydesc{##1}\glspostdescription
14696       \glstrprelocation ##2\tabularnewline
14697     }%
14698     \renewcommand{\subglossentry}[3]{%
14699       &
14700       \glssubentryitem{##2}%
14701       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription

```

```

14702     \glstrprelocation ##3\tabularnewline
14703 }%
14704 \ifglsnogroupskip
14705     \renewcommand*{\glsgroupskip}{}%
14706 \else
14707     \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
14708 \fi
14709 }
14710 }
14711 {}

```

Three column style:

```

14712 \ifcsdef{@glstyle@long3col}
14713 {%
14714     \renewglossarystyle{long3col}{%
14715         \renewenvironment{theglossary}%
14716             {\begin{longtable}\lp{\glstdescwidth}p{\glspagelistwidth}}}%
14717             {\end{longtable}}}%
14718     \renewcommand*{\glossaryheader}{}%
14719     \renewcommand*{\glsgroupheading}[1]{}%
14720     \renewcommand{\glossentry}[2]{%
14721         \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14722         \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
14723     }%
14724     \renewcommand{\subglossentry}[3]{%
14725         &
14726         \glssubentryitem{##2}%
14727         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14728         ##3\tabularnewline
14729     }%

```

Conditional needs to be outside of `\glsgroupskip` otherwise it can cause “Incomplete `\iftrue`” errors.

```

14730     \ifglsnogroupskip
14731         \renewcommand*{\glsgroupskip}{}%
14732     \else
14733         \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
14734     \fi
14735 }
14736 }
14737 {}

```

Four column style:

```

14738 \ifcsdef{@glstyle@long4col}
14739 {%
14740     \renewglossarystyle{long4col}{%
14741         \renewenvironment{theglossary}%
14742             {\begin{longtable}\llll}}}%
14743             {\end{longtable}}}%
14744     \renewcommand*{\glossaryheader}{}%
14745     \renewcommand*{\glsgroupheading}[1]{}%

```

```

14746 \renewcommand{\glossentry}[2]{%
14747   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14748   \glossentrydesc{##1}\glspostdescription &
14749   \glossentrysymbol{##1} &
14750   ##2\tabularnewline
14751 }%
14752 \renewcommand{\subglossentry}[3]{%
14753   &
14754   \glssubentryitem{##2}%
14755   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14756   \glossentrysymbol{##2} & ##3\tabularnewline
14757 }%

14758 \ifglsnogroupskip
14759   \renewcommand*\{\glsgroupskip}{}%
14760 \else
14761   \renewcommand*\{\glsgroupskip}{& & \tabularnewline}%
14762 \fi
14763 }
14764 }
14765 {}

```

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`.

```

14766 \ifcsdef{@glstyle@longragged}
14767 {%
14768   \renewglossarystyle{longragged}{%
14769     \renewenvironment{theglossary}%
14770       {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}}}%
14771       {\end{longtable}}%
14772     \renewcommand*\{\glossaryheader}{}%
14773     \renewcommand*\{\glsgroupheading}[1]{}%
14774     \renewcommand{\glossentry}[2]{%
14775       \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14776       \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
14777       \tabularnewline
14778     }%
14779     \renewcommand{\subglossentry}[3]{%
14780       &
14781       \glssubentryitem{##2}%
14782       \glstarget{##2}{\strut}\glossentrydesc{##2}%
14783       \glspostdescription\glxtrprelocation ##3%
14784       \tabularnewline

```

```

14785 }%
14786 \ifglsgroupskip
14787 \renewcommand*{\glsgroupskip}{}%
14788 \else
14789 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
14790 \fi
14791 }
14792 }
14793 {}

```

Three and four column styles don't use `\glstrprelocation` since the number list is in its own column.

```

14794 \ifcsdef{@glsstyle@longragged3col}
14795 {%
14796 \renewglossarystyle{longragged3col}{%
14797 \renewenvironment{theglossary}%
14798 {\begin{longtable}{l>\raggedright}p{\glsdescwidth}%
14799 >\raggedright}p{\glspagelistwidth}}}%
14800 {\end{longtable}}}%
14801 \renewcommand*{\glossaryheader}{}%
14802 \renewcommand*{\glsgroupheading}[1]{}%
14803 \renewcommand{\glossentry}[2]{%
14804 \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14805 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
14806 }%
14807 \renewcommand{\subglossentry}[3]{%
14808 &
14809 \glssubentryitem{##2}%
14810 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14811 ##3\tabularnewline
14812 }%
14813 \ifglsgroupskip
14814 \renewcommand*{\glsgroupskip}{}%
14815 \else
14816 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
14817 \fi
14818 }
14819 }
14820 {}

```

Four column style:

```

14821 \ifcsdef{@glsstyle@altlongragged4col}
14822 {%
14823 \renewglossarystyle{altlongragged4col}{%
14824 \renewenvironment{theglossary}%
14825 {\begin{longtable}{l>\raggedright}p{\glsdescwidth}l%
14826 >\raggedright}p{\glspagelistwidth}}}%
14827 {\end{longtable}}}%
14828 \renewcommand*{\glossaryheader}{}%

```

```

14829 \renewcommand*\glsgroupheading}[1]{}%
14830 \renewcommand{\glossentry}[2]{%
14831     \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14832     \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
14833     ##2\tabularnewline
14834 }%
14835 \renewcommand{\subglossentry}[3]{%
14836     &
14837     \glssubentryitem{##2}%
14838     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14839     \glossentrysymbol{##2} & ##3\tabularnewline
14840 }%

14841 \ifglsnogroupskip
14842     \renewcommand*\glsgroupskip{}{}%
14843 \else
14844     \renewcommand*\glsgroupskip{& & \tabularnewline}%
14845 \fi
14846 }
14847 }
14848 {}

```

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`.

```

14849 \ifcsdef{@glstyle@super}
14850 {%
14851     \renewglossarystyle{super}{%
14852         \renewenvironment{theglossary}%
14853             {\tablehead{}\tabletail{}}%
14854             \begin{supertabular}{\lp{\glsdescwidth}}%
14855             {\end{supertabular}}%
14856     \renewcommand*\glossaryheader{}{}%
14857     \renewcommand*\glsgroupheading}[1]{}%
14858     \renewcommand{\glossentry}[2]{%
14859         \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14860         \glossentrydesc{##1}\glspostdescription
14861         \glxtrprelocation ##2\tabularnewline
14862     }%
14863     \renewcommand{\subglossentry}[3]{%
14864         &
14865         \glssubentryitem{##2}%
14866         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
14867         \glxtrprelocation ##3\tabularnewline
14868     }%
14869     \ifglsnogroupskip
14870         \renewcommand*\glsgroupskip{}{}%

```



```

14871 \else
14872 \renewcommand*{\glsgroupskip}{& \tabularnewline}%
14873 \fi
14874 }
14875 }
14876 {}

```

Three column style:

```

14877 \ifcsdef{@glsstyle@super3col}
14878 {%
14879 \renewglossarystyle{super3col}{%
14880 \renewenvironment{theglossary}%
14881 {\tablehead{}\tabletail}%
14882 \begin{supertabular}{lp{\glstdescwidth}p{\glspagelistwidth}}}%
14883 {\end{supertabular}}%
14884 \renewcommand*{\glossaryheader}{}%
14885 \renewcommand*{\glsgroupheading}[1]{}%
14886 \renewcommand{\glossentry}[2]{%
14887 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14888 \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
14889 }%
14890 \renewcommand{\subglossentry}[3]{%
14891 &
14892 \glssubentryitem{##2}%
14893 \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14894 ##3\tabularnewline
14895 }%
14896 \ifglsnogroupskip
14897 \renewcommand*{\glsgroupskip}{}%
14898 \else
14899 \renewcommand*{\glsgroupskip}{& &\tabularnewline}%
14900 \fi
14901 }
14902 }
14903 {}

```

Four column styles:

```

14904 \ifcsdef{@glsstyle@super4col}
14905 {%
14906 \renewglossarystyle{super4col}{%
14907 \renewenvironment{theglossary}%
14908 {\tablehead{}\tabletail}%
14909 \begin{supertabular}{llll}}{%
14910 \end{supertabular}}%
14911 \renewcommand*{\glossaryheader}{}%
14912 \renewcommand*{\glsgroupheading}[1]{}%
14913 \renewcommand{\glossentry}[2]{%
14914 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14915 \glossentrydesc{##1}\glspostdescription &

```

```

14916     \glossentrysymbol{##1} & ##2\tabularnewline
14917 }%
14918 \renewcommand{\subglossentry}[3]{%
14919     &
14920     \glssubentryitem{##2}%
14921     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14922     \glossentrysymbol{##2} & ##3\tabularnewline
14923 }%

14924 \ifglsnogroupskip
14925     \renewcommand*{\glsgroupskip}{}%
14926 \else
14927     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
14928 \fi
14929 }
14930 }
14931 {}

```

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 `\glxtrprelocation`.

```

14932 \ifcsdef{@glstyle@superragged}
14933 {%
14934     \renewglossarystyle{superragged}{%
14935         \renewenvironment{theglossary}%
14936             {\tablehead{}\tabletail}%
14937             \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
14938             {\end{supertabular}}%
14939     \renewcommand*{\glossaryheader}{}%
14940     \renewcommand*{\glsgroupheading}[1]{}%
14941     \renewcommand{\glossentry}[2]{%
14942         \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14943         \glossentrydesc{##1}\glspostdescription\glxtrprelocation ##2%
14944         \tabularnewline
14945     }%
14946     \renewcommand{\subglossentry}[3]{%
14947         &
14948         \glssubentryitem{##2}%
14949         \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription
14950         \glxtrprelocation ##3%
14951         \tabularnewline
14952     }%
14953     \ifglsnogroupskip
14954         \renewcommand*{\glsgroupskip}{}%
14955     \else
14956         \renewcommand*{\glsgroupskip}{& \tabularnewline}%

```

```

14957   \fi
14958 }
14959 }
14960 {}

```

Three column style:

```

14961 \ifcsdef{@glsstyle@superragged3col}
14962 {%
14963   \renewglossarystyle{superragged3col}{%
14964     \renewenvironment{theglossary}%
14965       {\tablehead{}\tabletail{}}%
14966       \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
14967         >{\raggedright}p{\glspagelistwidth}}}%
14968       {\end{supertabular}}%
14969   \renewcommand*{\glossaryheader}{}%
14970   \renewcommand*{\glsgroupheading}[1]{}%
14971   \renewcommand{\glossentry}[2]{%
14972     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
14973     \glossentrydesc{##1}\glspostdescription &
14974     ##2\tabularnewline
14975   }%
14976   \renewcommand{\subglossentry}[3]{%
14977     &
14978     \glssubentryitem{##2}%
14979     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
14980     ##3\tabularnewline
14981   }%
14982   \ifglsnogroupskip
14983     \renewcommand*{\glsgroupskip}{}%
14984   \else
14985     \renewcommand*{\glsgroupskip}{ & &\tabularnewline}%
14986   \fi
14987 }
14988 }
14989 {}

```

Four columns:

```

14990 \ifcsdef{@glsstyle@altsuperragged4col}
14991 {%
14992   \renewglossarystyle{altsuperragged4col}{%
14993     \renewenvironment{theglossary}%
14994       {\tablehead{}\tabletail{}}%
14995       \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
14996         >{\raggedright}p{\glspagelistwidth}}}%
14997       {\end{supertabular}}%
14998   \renewcommand*{\glossaryheader}{}%
14999   \renewcommand{\glossentry}[2]{%
15000     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
15001     \glossentrydesc{##1}\glspostdescription &

```

```

15002     \glossentrysymbol{##1} & ##2\tabularnewline
15003 }%
15004 \renewcommand{\subglossentry}[3]{%
15005     &
15006     \glssubentryitem{##2}%
15007     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
15008     \glossentrysymbol{##2} & ##3\tabularnewline
15009 }%

15010 \ifglsnogroupskip
15011     \renewcommand*{\glsgroupskip}{}%
15012 \else
15013     \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
15014 \fi
15015 }
15016 }
15017 {}

```

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`.

```

15018 \ifdef{\@glsstyle@inline}
15019 {%
15020     \renewcommand*{\glspostinline}{.\spacefactor\sfcode{.}}
15021     Just use \glstrpostdescription instead of \glspostdescription.
15022     \renewcommand*{\glsinlinedescformat}[3]{%
15023         \space#1\glstrpostdescription}
15024     \renewcommand*{\glsinlinesubdescformat}[3]{%
15025         #1\glstrpostdescription}

```

The default settings don't show the location lists, so there's no adjustment for `\glstrprelocation`.

```

15025 }
15026 {}

```

2.8 Tree Styles

Redefine both `\glstreenamfmt` and `\glstreegroupheaderfmt` in terms of `\glstreedefaultnamfmt` to make it easier to change both at the same time or only change one without affecting the other.

```

15027 \ifdef\glstreenamfmt
15028 {
15029     \renewcommand{\glstreenamfmt}[1]{\textbf{#1}}
15030 }
15031 \ifdef\glstreegroupheaderfmt
15032 {
15033     \renewcommand{\glstreegroupheaderfmt}[1]{\textbf{#1}}
15034 }
15035 \ifdef\glstreedefaultnamfmt
15036 {
15037     \renewcommand{\glstreedefaultnamfmt}[1]{\textbf{#1}}
15038 }

```

`\glstreenamefmt`

```
15030 \renewcommand{\glstreenamefmt}[1]{\glstreedefaultnamefmt{#1}}
```

`\glstreegroupheaderfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
15031 \def\glstreegroupheaderfmt#1{\glstreedefaultnamefmt{#1}}
```

`\glstreenavigationfmt` This command was only introduced to glossary-tree v4.22, so it may not be defined.

```
15032 \def\glstreenavigationfmt#1{\glstreedefaultnamefmt{#1}}
```

`\glstreePreHeader` Takes the label as the first argument and title as the second argument so this can be modified to add a bookmark.

```
15033 \newcommand{\glstreePreHeader}[2]{}

15034 }
15035 {}
```

The index style is redefined so that the space before the number list isn't hard coded.

```
15036 \ifdef{\@glsstyle@index}
15037 {
```

`\glstreeprelocation` The space before the number list for top-level entries. This is shared by the other tree styles.

```
15038 \newcommand*{\glstreeprelocation}{\glstxtrprelocation}
```

`\glstreechildprelocation` The space before the number list for child entries. This is shared by the other tree styles.

```
15039 \newcommand*{\glstreechildprelocation}{\glstreeprelocation}
```

Don't prohibit a page break at the start of a new group if there's no header.

`\glstreegroupskip`

```
15040 \newcommand{\glstreegroupskip}{\indexspace}
```

`\glstreegroupheaderskip` This doesn't include `\@afterheading` as it can cause interference with some styles.

```
15041 \newcommand{\glstreegroupheaderskip}{\nopagebreak\glstreegroupskip\nobreak}
```

Modify the index style.

```
15042 \renewglossarystyle{index}{%
15043   \renewenvironment{theglossary}%
15044     {\setlength{\parindent}{0pt}%
15045      \setlength{\parskip}{0pt plus 0.3pt}%
15046      \let\item\glstreeitem
15047      \let\subitem\glstreesubitem
15048      \let\subsubitem\glstreesubsubitem
15049     }%
15050   {\par}%
15051   \renewcommand*{\glossaryheader}{}%
15052   \renewcommand*{\glsgroupheading}[1]{}%
15053   \renewcommand*{\glossentry}[2]{%
15054     \item\glsentryitem{#1}%
```

```

15055     \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
15056     \glstreesymbol{##1}%
15057     \glstreeDescLoc{##1}{##2}%
15058 }%
15059 \renewcommand{\subglossentry}[3]{%
15060     \ifcase##1\relax
15061     \item
15062     \or
15063     \subitem
15064     \glssubentryitem{##2}%
15065     \else
15066     \subsubitem
15067     \fi
15068     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
15069     \glstreechildsymbol{##2}%
15070     \glstreeChildDescLoc{##2}{##3}%
15071 }%
15072 \renewcommand*\glsgroupskip{\ifglsnogroupskip\else\glstreegroupskip\fi}%
15073 }
15074 }
15075 {}

```

The indexgroup style is redefined to discourage a page break after the heading.

```

15076 \ifdef{\@glsstyle@indexgroup}
15077 {%
15078     \renewglossarystyle{indexgroup}{%
15079         \setglossarystyle{index}%
15080         \renewcommand*\glsgroupheading[1]{%
15081             \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15082             \glstreePreHeader{##1}{\glsxtr@grptitle}%
15083             \item\glstreegroupheaderfmt{\glsxtr@grptitle}%
15084             \glstreegroupheaderskip\@afterheading
15085         }%
15086     }
15087 }
15088 {}

```

Similarly for indexhypergroup.

```

15089 \ifdef{\@glsstyle@indexhypergroup}
15090 {%
15091     \renewglossarystyle{indexhypergroup}{%
15092         \setglossarystyle{index}%
15093         \renewcommand*\glossaryheader{%
15094             \item\glstreenavigationfmt{\glsnavigation}%
15095             \glstreegroupheaderskip\@afterheading}%
15096         \renewcommand*\glsgroupheading[1]{%
15097             \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15098             \glstreePreHeader{##1}{\glsxtr@grptitle}%
15099             \item\glstreegroupheaderfmt
15100                 {\glsnahypertarget{##1}{\glsxtr@grptitle}}%

```

```

15101      \glstreegroupheaderskip\@afterheading}%
15102    }%
15103  }
15104 {}

```

Adjust tree style to remove hard coded space before number list.

```

15105 \ifdef{\@glsstyle@tree}
15106 {%

```

Provide a command for use with the tree styles that displays the pre-description separator, the description and post-description hook.

`\glstreedesc`

```

15107 \newcommand{\glstreedesc}[1]{%
15108   \glstreepredesc\glossentrydesc{#1}\glspostdescription
15109 }

```

`\glstreeDescLoc`

```
\glstreeDescLoc{<label>}{<location>}
```

This checks for the description and symbol. If both are missing, a different separator may be required. For example, a comma and space if there's no description or symbol but just a space if either of those fields are present.

```

15110 \newcommand{\glstreeDescLoc}[2]{%
15111   \ifglshasdesc{#1}%
15112   {\glstreedesc{#1}\glstreeprelocation}%
15113   {\ifglshassymbol{#1}{\glstreeprelocation}{\glstreeNoDescSymbolPreLocation}}}%
15114   #2%
15115 }

```

`\glstreeNoDescSymbolPreLocation`

```
\glstreeNoDescSymbolPreLocation
```

```
15116 \newcommand{\glstreeNoDescSymbolPreLocation}{\space}
```

Similarly for the symbol.

`\glstreesymbol`

```

15117 \newcommand{\glstreesymbol}[1]{%
15118   \ifglshassymbol{#1}{\space(\glossentrysymbol{#1})}{}%
15119 }%

```

And for the child entries:

lstreechilddesc

```
15120 \newcommand{\glstreechilddesc}[1]{%
15121 \glstreechildpredesc\glossentrydesc{#1}\glspostdescription
15122 }%
```

reeChildDescLoc

```
15123 \newcommand{\glstreeChildDescLoc}[2]{%
15124 \ifglshasdesc{#1}%
15125 {\glstreechilddesc{#1}\glstreechildprelocation}%
15126 {\ifglshassymbol{#1}{\glstreechildprelocation}%
15127 {\glstreeNoDescSymbolPreLocation}%
15128 }%
15129 #2%
15130 }%
```

treechildsymbol This just behaves in the same way as the top-level.

```
15131 \newcommand{\glstreechildsymbol}[1]{%
15132 \glstreesymbol{#1}%
15133 }%

15134 \renewglossarystyle{tree}{%
15135 \renewenvironment{theglossary}%
15136 {\setlength{\parindent}{0pt}%
15137 \setlength{\parskip}{0pt plus 0.3pt}}%
15138 {}%
15139 \renewcommand*\{\glossaryheader}{}%
15140 \renewcommand*\{\glsgroupheading}[1]{}%
15141 \renewcommand{\glossentry}[2]{%
15142 \hangindent0pt\relax
15143 \parindent0pt\relax
15144 \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
15145 \glstreesymbol{##1}%
15146 \glstreeDescLoc{##1}{##2}\par
15147 }%
15148 \renewcommand{\subglossentry}[3]{%
15149 \hangindent##1\glstreeindent\relax
15150 \parindent##1\glstreeindent\relax
15151 \ifnum##1=1\relax
15152 \glssubentryitem{##2}%
15153 \fi
15154 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
15155 \glstreechildsymbol{##2}%
15156 \glstreeChildDescLoc{##2}{##3}\par
15157 }%
15158 \renewcommand*\{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
15159 }%
15160 }
15161 {}
```


The treegroup style is redefined to discourage a page break after the heading.

```

15162 \ifdef{\@glsstyle@treegroup}
15163 {%
15164   \renewglossarystyle{treegroup}{%
15165     \setglossarystyle{tree}%
15166     \renewcommand{\glsgroupheading}[1]{%
15167       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15168       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15169       \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
15170       \glstreegroupheaderskip\@afterheading}%
15171   }
15172 }
15173 {}

```

Similarly for treehypergroup

```

15174 \ifdef{\@glsstyle@treehypergroup}
15175 {%
15176   \renewglossarystyle{treehypergroup}{%
15177     \setglossarystyle{tree}%
15178     \renewcommand*\glossaryheader{%
15179       \par\noindent\glstreenavigationfmt{\glsnavigation}%
15180       \glstreegroupheaderskip\@afterheading}%
15181     \renewcommand*\glsgroupheading[1]{%
15182       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15183       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15184       \par\noindent
15185       \glstreegroupheaderfmt
15186       {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15187       \glstreegroupheaderskip\@afterheading}%
15188   }
15189 }
15190 {}

```

Adjust treenoname style to remove hard coded space before number list.

```

15191 \ifdef{\@glsstyle@treenoname}
15192 {%

```

Provide a command for use with the treenoname styles that displays the pre-description separator, the description and post-description hook.

streenonamedesc

```

15193   \newcommand{\glstreenonamedesc}[1]{%
15194     \glstreepredesc\glossentrydesc{##1}\glspostdescription
15195   }%

```

Similarly for the symbol.

reenonamesymbol

```

15196   \newcommand{\glstreenonamesymbol}[1]{%
15197     \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
15198   }%

```

nonamechilddesc The child entry doesn't have the pre-description separator as the name isn't displayed.

```

15199 \newcommand{\glstreenonamechilddesc}[1]{%
15200   \glossentrydesc{#1}\glspostdescription
15201 }%

15202 \renewglossarystyle{treenoname}{%
15203   \renewenvironment{theglossary}%
15204     {\setlength{\parindent}{0pt}%
15205     \setlength{\parskip}{0pt plus 0.3pt}}%
15206   {%
15207     \renewcommand*{\glossaryheader}{}%
15208     \renewcommand*{\glsgroupheading}[1]{}%
15209     \renewcommand{\glossentry}[2]{%
15210       \hangindent0pt\relax
15211       \parindent0pt\relax
15212       \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
15213       \glstreenonamesymbol{##1}%
15214       \glstreenonamedesc{##1}%
15215       \glstreeDescLoc{##1}{##2}\par
15216     }%
15217     \renewcommand{\subglossentry}[3]{%
15218       \hangindent##1\glstreeindent\relax
15219       \parindent##1\glstreeindent\relax
15220       \ifnum##1=1\relax
15221         \glssubentryitem{##2}%
15222       \fi
15223       \glstarget{##2}{\strut}%
15224       \glstreenonamechilddesc{##2}%
15225       \glstreechildprelocation##3\par
15226     }%
15227     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
15228   }
15229 }
15230 {}

```

The treenonamegroup style is redefined to discourage a page break after the heading.

```

15231 \ifdef{\@glssstyle@treenonamegroup}
15232 {%
15233   \renewglossarystyle{treenonamegroup}{%
15234     \setglossarystyle{treenoname}%
15235     \renewcommand{\glsgroupheading}[1]{%
15236       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15237       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15238       \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
15239       \glstreegroupheaderskip\@afterheading
15240     }%
15241   }
15242 }
15243 {}

```

Similarly for `treenonamehypergroup`

```

15244 \ifdef{\@glsstyle@treenonamehypergroup}
15245 {%
15246   \renewglossarystyle{treenonamehypergroup}{%
15247     \setglossarystyle{treenoname}%
15248     \renewcommand*{\glossaryheader}{%
15249       \par\noindent\glstreenavigationfmt{\glsnavigation}%
15250       \glstreegroupheaderskip\@afterheading}%
15251     \renewcommand*{\glsgroupheading}[1]{%
15252       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15253       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15254       \par\noindent
15255       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15256       \glstreegroupheaderskip\@afterheading}%
15257   }
15258 }
15259 {}

```

The `almtree` style is redefined to make it easier to made minor adjustments.

```

15260 \ifdef{\@glsstyle@almtree}
15261 {%

```

Only redefine this style if it's already been defined.

`bolDescLocation`

`\glsxtralmtreeSymbolDescLocation{<label>}{<location list>}`

Layout the symbol, description and location for top-level entries.

```

15262 \newcommand{\glsxtralmtreeSymbolDescLocation}[2]{%
15263   {%
15264     \let\par\glsxtrAltTreePar
15265     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
15266     \ifglshasdesc{#1}{\glossentrydesc{#1}\glspostdescription}{}%
15267     \glstreeDescLoc{#1}{#2}\par
15268   }%
15269 }

```

`trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

15270 \newlength\glsxtrAltTreeIndent

```

`lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

15271 \newcommand{\glsxtrAltTreePar}{%
15272   \@@par
15273   \glsxtrAltTreeSetHangIndent
15274   \setlength{\parindent}{\dimexpr\hangindent+\glsxtrAltTreeIndent}%
15275 }

```

bolDescLocation

`\glxtralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```
15276 \newcommand{\glxtralttreeSubSymbolDescLocation}[3]{%
15277   \glxtralttreeSymbolDescLocation{#2}{#3}%
15278 }
```

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.

```
15279 \newlength\glxtrreetopindent
```

sxtralttreeInit User-level initialisation for the alttree style.

```
15280 \newcommand*{\glxtralttreeInit}{%
15281   \settowidth{\glxtrreetopindent}{\glstreenamefmt{\glsgetwidestname\space}}%
15282   \glxtrAltTreeIndent=\parindent
15283 }
```

\gglsetwidest The original \glsetwidest only uses \def. This uses \gdef.

```
15284 \newcommand*{\gglsetwidest}[2][0]{%
15285   \csgdef{@glswidestname\romannumeral#1}{#2}%
15286 }
```

\eglssetwidest The original \glsetwidest only uses \def. This uses \protected@csedef.

```
15287 \newcommand*{\eglssetwidest}[2][0]{%
15288   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
15289 }
```

\xglsetwidest Like the above but uses \protected@csxdef.

```
15290 \newcommand*{\xglsetwidest}[2][0]{%
15291   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
15292 }
```

glupdatewidest Only sets if new value is wider than old value.

```
15293 \newcommand*{\glupdatewidest}[2][0]{%
15294   \ifcsundef{@glswidestname\romannumeral#1}%
15295   {\csdef{@glswidestname\romannumeral#1}{#2}}%
15296   {%
15297     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
15298     \settowidth{\dimen@ii}{#2}%
15299     \ifdim\dimen@ii>\dimen@
15300     \csdef{@glswidestname\romannumeral#1}{#2}%
15301     \fi
15302   }%
15303 }
```

glsupdatewidest As above but global definition.

```
15304 \newcommand*{\glsupdatewidest}[2][0]{%
15305   \ifcsundef{@glswidestname\romannumeral#1}%
15306   {\csgdef{@glswidestname\romannumeral#1}{#2}}%
15307   {%
15308     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
15309     \settowidth{\dimen@ii}{#2}%
15310     \ifdim\dimen@ii>\dimen@
15311     \csgdef{@glswidestname\romannumeral#1}{#2}%
15312     \fi
15313   }%
15314 }
```

glsupdatewidest As \glsupdatewidest but expands value.

```
15315 \newcommand*{\eglsupdatewidest}[2][0]{%
15316   \ifcsundef{@glswidestname\romannumeral#1}%
15317   {\protected@csedef{@glswidestname\romannumeral#1}{#2}}%
15318   {%
15319     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
15320     \settowidth{\dimen@ii}{#2}%
15321     \ifdim\dimen@ii>\dimen@
15322     \protected@csedef{@glswidestname\romannumeral#1}{#2}%
15323     \fi
15324   }%
15325 }
```

glsupdatewidest As above but global.

```
15326 \newcommand*{\xglsupdatewidest}[2][0]{%
15327   \ifcsundef{@glswidestname\romannumeral#1}%
15328   {\protected@csxdef{@glswidestname\romannumeral#1}{#2}}%
15329   {%
15330     \settowidth{\dimen@}{\csuse{@glswidestname\romannumeral#1}}%
15331     \settowidth{\dimen@ii}{#2}%
15332     \ifdim\dimen@ii>\dimen@
15333     \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
15334     \fi
15335   }%
15336 }
```

glsgetwidestname Provide a user-level macro to obtain the widest top-level name.

```
15337 \newcommand*{\glsgetwidestname}{\@glswidestname}
```

glsgetwidestsubname Provide a user-level macro to obtain the widest sub-entry name.

```
15338 \newcommand*{\glsgetwidestsubname}[1]{%
15339   \ifcsundef{@glswidestname\romannumeral#1}%
15340   {\@glswidestname}%
15341   {\csuse{@glswidestname\romannumeral#1}}%
15342 }
```

`estTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```
15343 \let\glsFindWidestTopLevelName\glsfindwidesttoplevelname
```

`sedTopLevelName` Like `\glsfindwidesttoplevelname` but has an additional check that the entry has been used. Only useful if the glossaries occur at the end of the document, in which case this command should go at the start of the glossary. Alternatively, place at the end of the document and save for the next run.

```
15344 \newrobustcmd*{\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
15345   \dimen@=0pt\relax
15346   \gls@tmplen=0pt\relax
15347   \foralllglossaries[#1]{\@gls@type}%
15348   {%
15349     \forlgsentries[\@gls@type]{\@glo@label}%
15350     {%
15351       \ifglused{\@glo@label}%
15352       {%
15353         \ifglshasparent{\@glo@label}%
15354         {}%
15355         {%
15356           \settowidth{\dimen@}%
15357             {\glstreenamefmt{\glstryname{\@glo@label}}}%
15358           \ifdim\dimen@>\gls@tmplen
15359             \gls@tmplen=\dimen@
15360             \eglssetwidest{\glstryname{\@glo@label}}%
15361           \fi
15362         }%
15363       }%
15364     }%
15365   }%
15366 }%
15367 }
```

`destUsedAnyName` Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```
15368 \newrobustcmd*{\glsFindWidestUsedAnyName}[1][\@glo@types]{%
15369   \dimen@=0pt\relax
15370   \gls@tmplen=0pt\relax
15371   \foralllglossaries[#1]{\@gls@type}%
15372   {%
15373     \forlgsentries[\@gls@type]{\@glo@label}%
15374     {%
15375       \ifglused{\@glo@label}%
15376       {%
15377         \settowidth{\dimen@}%
15378           {\glstreenamefmt{\glstryname{\@glo@label}}}%
15379         \ifdim\dimen@>\gls@tmplen
15380           \gls@tmplen=\dimen@
15381           \eglssetwidest{\glstryname{\@glo@label}}%

```

```

15382         \fi
15383     }%
15384     {}%
15385     }%
15386     }%
15387 }

```

ndWidestAnyName Like the above but doesn't check if the entry has been used.

```

15388 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
15389     \dimen@=0pt\relax
15390     \gls@tmplen=0pt\relax
15391     \forallglossaries[#1]{\@gls@type}%
15392     {%
15393         \forglsentries[\@gls@type]{\@glo@label}%
15394         {%
15395             \settowidth{\dimen@}%
15396             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
15397             \ifdim\dimen@>\gls@tmplen
15398                 \gls@tmplen=\dimen@
15399                 \eglssetwidest{\glsentryname{\@glo@label}}%
15400         }%
15401     }%
15402 }%
15403 }

```

estUsedLevelTwo This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well.

Any entry that has a great-grandparent is ignored.

```

15404 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
15405     \dimen@=0pt\relax
15406     \dimen@i=0pt\relax
15407     \dimen@ii=0pt\relax
15408     \forallglossaries[#1]{\@gls@type}%
15409     {%
15410         \forglsentries[\@gls@type]{\@glo@label}%
15411         {%
15412             \ifglssused{\@glo@label}%
15413             {%
15414                 \ifglshasparent{\@glo@label}%
15415                 {%
15416                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
15417                     \ifglshasparent{\@glo@parent}%
15418                     {%
15419                         \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
15420                     \ifglshasparent{\@glo@parent}%
15421                     {}%
15422                 }%
15423                 \settowidth{\gls@tmplen}%
15424                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
15425                 \ifdim\gls@tmplen>\dimen@ii

```

```

15426         \dimen@ii=\gls@tmplen
15427         \eglssetwidest[2]{\glsentryname{\@glo@label}}}%
15428     \fi
15429 }%
15430 }%
15431 {%
15432     \settowidth{\gls@tmplen}%
15433         {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
15434     \ifdim\gls@tmplen>\dimen@i
15435         \dimen@i=\gls@tmplen
15436         \eglssetwidest[1]{\glsentryname{\@glo@label}}}%
15437     \fi
15438 }%
15439 }%
15440 {%
15441     \settowidth{\gls@tmplen}%
15442         {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
15443     \ifdim\gls@tmplen>\dimen@
15444         \dimen@=\gls@tmplen
15445         \eglssetwidest{\glsentryname{\@glo@label}}}%
15446     \fi
15447 }%
15448 }%
15449 {}%
15450 }%
15451 }%
15452 }

```

dWidestLevelTwo This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

15453 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
15454     \dimen@=0pt\relax
15455     \dimen@i=0pt\relax
15456     \dimen@ii=0pt\relax
15457     \foralllglossaries[#1]{\@gls@type}%
15458     {%
15459         \forglsentries[\@gls@type]{\@glo@label}%
15460         {%
15461             \ifglshasparent{\@glo@label}%
15462             {%
15463                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@label}@parent}}%
15464                 \ifglshasparent{\@glo@parent}%
15465                 {%
15466                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel}{\@glo@parent}@parent}}%
15467                     \ifglshasparent{\@glo@parent}%
15468                     {}%
15469                 }%
15470                 \settowidth{\gls@tmplen}%
15471                     {\glsstreenamfmt{\glsentryname{\@glo@label}}}%
15472                 \ifdim\gls@tmplen>\dimen@ii

```



```

15473         \dimen@ii=\gls@tmplen
15474         \eglssetwidest[2]{\glsentryname{\@glo@label}}}%
15475     \fi
15476 }%
15477 }%
15478 {%
15479     \settowidth{\gls@tmplen}%
15480     {\glstreenamefmt{\glsentryname{\@glo@label}}}%
15481     \ifdim\gls@tmplen>\dimen@i
15482         \dimen@i=\gls@tmplen
15483         \eglssetwidest[1]{\glsentryname{\@glo@label}}}%
15484     \fi
15485 }%
15486 }%
15487 {%
15488     \settowidth{\gls@tmplen}%
15489     {\glstreenamefmt{\glsentryname{\@glo@label}}}%
15490     \ifdim\gls@tmplen>\dimen@
15491         \dimen@=\gls@tmplen
15492         \eglssetwidest{\glsentryname{\@glo@label}}}%
15493     \fi
15494 }%
15495 }%
15496 }%
15497 }

```

`\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.

```

15498 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
15499     \dimen@=0pt\relax
15500     \gls@tmplen=0pt\relax
15501     #2=0pt\relax
15502     \foralllglossaries[#1]{\@gls@type}%
15503     {%
15504         \forglsentries[\@gls@type]{\@glo@label}%
15505         {%
15506             \ifglsused{\@glo@label}%
15507             {%
15508                 \settowidth{\dimen@}%
15509                 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
15510                 \ifdim\dimen@>\gls@tmplen
15511                     \gls@tmplen=\dimen@
15512                     \eglssetwidest{\glsentryname{\@glo@label}}}%
15513             \fi
15514             \settowidth{\dimen@}%
15515             {\glsentrysymbol{\@glo@label}}}%
15516         \ifdim\dimen@>#2\relax
15517             #2=\dimen@
15518         \fi

```

```

15519     }%
15520     {}%
15521     }%
15522     }%
15523 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

15524 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
15525     \dimen@=0pt\relax
15526     \gls@tmplen=0pt\relax
15527     #2=0pt\relax
15528     \forallglossaries[#1]{\@gls@type}%
15529     {%
15530         \forlgsentries[\@gls@type]{\@glo@label}%
15531         {%
15532             \settowidth{\dimen@}%
15533             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
15534             \ifdim\dimen@>\gls@tmplen
15535                 \gls@tmplen=\dimen@
15536                 \eglissetwidest{\glsentryname{\@glo@label}}%
15537             \fi
15538             \settowidth{\dimen@}%
15539             {\glsentrysymbol{\@glo@label}}%
15540             \ifdim\dimen@>#2\relax
15541                 #2=\dimen@
15542             \fi
15543         }%
15544     }%
15545 }

```

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.

```

15546 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
15547     \dimen@=0pt\relax
15548     \gls@tmplen=0pt\relax
15549     #2=0pt\relax
15550     #3=0pt\relax
15551     \forallglossaries[#1]{\@gls@type}%
15552     {%
15553         \forlgsentries[\@gls@type]{\@glo@label}%
15554         {%
15555             \ifglsused{\@glo@label}%
15556             {%
15557                 \settowidth{\dimen@}%
15558                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
15559                 \ifdim\dimen@>\gls@tmplen
15560                     \gls@tmplen=\dimen@

```

```

15561         \eglssetwidest{\glsentryname{\@glo@label}}%
15562     \fi
15563     \settowidth{\dimen@}%
15564     {\glsentrysymbol{\@glo@label}}%
15565     \ifdim\dimen@>#2\relax
15566         #2=\dimen@
15567     \fi
15568     \settowidth{\dimen@}%
15569     {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
15570     \ifdim\dimen@>#3\relax
15571         #3=\dimen@
15572     \fi
15573 }%
15574 {}%
15575 }%
15576 }%
15577 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

15578 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
15579     \dimen@=0pt\relax
15580     \gls@tmplen=0pt\relax
15581     #2=0pt\relax
15582     #3=0pt\relax
15583     \forallglossaries[#1]{\@gls@type}%
15584     {%
15585         \forglsentries[\@gls@type]{\@glo@label}%
15586         {%
15587             \settowidth{\dimen@}%
15588             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
15589             \ifdim\dimen@>\gls@tmplen
15590                 \gls@tmplen=\dimen@
15591                 \eglssetwidest{\glsentryname{\@glo@label}}%
15592             \fi
15593             \settowidth{\dimen@}%
15594             {\glsentrysymbol{\@glo@label}}%
15595             \ifdim\dimen@>#2\relax
15596                 #2=\dimen@
15597             \fi
15598             \settowidth{\dimen@}%
15599             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
15600             \ifdim\dimen@>#3\relax
15601                 #3=\dimen@
15602             \fi
15603         }%
15604     }%
15605 }

```

AnyNameLocation Like the `\glsFindWidestUsedAnyNameSymbolLocation` but doesn't measure the symbol.

The length of the widest location list is stored in the second argument, which should be a length register.

```

15606 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
15607   \dimen@=0pt\relax
15608   \gls@tmplen=0pt\relax
15609   #2=0pt\relax
15610   \forallglossaries[#1]{\@gls@type}%
15611   {%
15612     \forallglsentries[\@gls@type]{\@glo@label}%
15613     {%
15614       \ifglsused{\@glo@label}%
15615       {%
15616         \settowidth{\dimen@}%
15617           {\glstreenamfmt{\glsentryname{\@glo@label}}}%
15618         \ifdim\dimen@>\gls@tmplen
15619           \gls@tmplen=\dimen@
15620           \eglssetwidest{\glsentryname{\@glo@label}}%
15621         \fi
15622         \settowidth{\dimen@}%
15623           {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
15624         \ifdim\dimen@>#2\relax
15625           #2=\dimen@
15626         \fi
15627       }%
15628     }%
15629   }%
15630 }%
15631 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

15632 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
15633   \dimen@=0pt\relax
15634   \gls@tmplen=0pt\relax
15635   #2=0pt\relax
15636   \forallglossaries[#1]{\@gls@type}%
15637   {%
15638     \forallglsentries[\@gls@type]{\@glo@label}%
15639     {%
15640       \settowidth{\dimen@}%
15641         {\glstreenamfmt{\glsentryname{\@glo@label}}}%
15642       \ifdim\dimen@>\gls@tmplen
15643         \gls@tmplen=\dimen@
15644         \eglssetwidest{\glsentryname{\@glo@label}}%
15645       \fi
15646       \settowidth{\dimen@}%
15647         {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
15648       \ifdim\dimen@>#2\relax
15649         #2=\dimen@
15650       \fi

```

```

15651     }%
15652   }%
15653 }

```

`\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`.

```

15654 \newcommand*{\glstxtrComputeTreeIndent}[1]{%
15655   \glstreeindent=\glstxtrtreetopindent\relax
15656 }

```

`\glstxtrComputeTreeSubIndent`

`\glstxtrComputeTreeSubIndent{<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.

```

15657 \newcommand*{\glstxtrComputeTreeSubIndent}[3]{%
15658   \ifcsundef{@glswidestname\romannumeral#1}%
15659   {%
15660     \settowidth{#3}{\glstreenamfmt{\@glswidestname\space}}%
15661   }%
15662   {%
15663     \settowidth{#3}{\glstreenamfmt{%
15664       \csname @glswidestname\romannumeral#1\endcsname\space}}%
15665   }%
15666 }

```

`\glstxtrAltTreeSetHangIndent` Set `\hangindent` for top-level entries:

```

15667 \newcommand*{\glstxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}

```

`\glstxtrAltTreeSetSubHangIndent` Set `\hangindent` for sub-entries:

```

15668 \newcommand*{\glstxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}

```

Redefine `\alttree`:

```

15669 \renewglossarystyle{alttree}{%
15670   \renewenvironment{theglossary}%
15671   {%
15672     \glstxtralttreeInit
15673     \def\@gls@prevlevel{-1}%
15674     \mbox{}\par}%
15675   {\par}%
15676   \renewcommand*{\glossaryheader}{}%
15677   \renewcommand*{\glsgroupheading}[1]{}%
15678   \renewcommand{\glossentry}[2]{%
15679     \ifnum\@gls@prevlevel=0\relax

```

```

15680 \else
15681 \glxstrComputeTreeIndent{##1}%
15682 \fi
15683 \parindent\glstreeindent
15684 \glxstrAltTreeSetHangIndent
15685 \makebox[Opt][r]%
15686 {%
15687 \glstreenamebox{\glstreeindent}%
15688 {%
15689 \glssentryitem{##1}%
15690 \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
15691 }%
15692 }%
15693 \glxstralttreeSymbolDescLocation{##1}{##2}%
15694 \def\@gls@prevlevel{0}%
15695 }
15696 \renewcommand{\subglossentry}[3]{%
15697 \ifnum##1=1\relax
15698 \glssubentryitem{##2}%
15699 \fi
15700 \ifnum\@gls@prevlevel=##1\relax
15701 \else
15702 \glxstrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
15703 \ifnum\@gls@prevlevel<##1\relax
15704 \setlength\glstreeindent\gls@tmplen
15705 \addtolength\glstreeindent\parindent
15706 \parindent\glstreeindent
15707 \else
15708 \ifnum\@gls@prevlevel=0\relax
15709 \glxstrComputeTreeIndent{##2}%
15710 \else
15711 \glxstrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
15712 \fi
15713 \addtolength\parindent{-\glstreeindent}%
15714 \setlength\glstreeindent\parindent
15715 \fi
15716 \fi
15717 \glxstrAltTreeSetSubHangIndent{##1}%
15718 \makebox[Opt][r]{\glstreenamebox{\gls@tmplen}{%
15719 \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
15720 \glxstralttreeSubSymbolDescLocation{##1}{##2}{##3}%
15721 \def\@gls@prevlevel{##1}%
15722 }%
15723 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\glstreegroupskip\fi}%
15724 }
15725 }%
15726 {%
15727 }

```

Redefine `alttreegroup` so that it discourages a break after group headings.

```

15728 \ifdef{\@glsstyle@almtreegroup}
15729 {%
15730   \renewglossarystyle{almtreegroup}{%
15731     \setglossarystyle{almtree}%
15732     \renewcommand{\glsgroupheading}[1]{\par
15733       \def\@gls@prevlevel{-1}%
15734       \hangindentOpt\relax
15735       \parindentOpt\relax
15736       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15737       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15738       \glstreegroupheaderfmt{\glsxtr@grptitle}%

```

Can't use \@afterheading here as it messes with the first item of the group.

```

15739   \glstreegroupheaderskip
15740   }%
15741 }%
15742 }%
15743 {%
15744 }

```

Similarly for almtreehypergroup.

```

15745 \ifdef{\@glsstyle@almtreehypergroup}
15746 {%
15747   \renewglossarystyle{almtreehypergroup}{%
15748     \setglossarystyle{almtree}%
15749     \renewcommand*\{glossaryheader}{%
15750       \par
15751       \def\@gls@prevlevel{-1}%
15752       \hangindentOpt\relax
15753       \parindentOpt\relax
15754       \glstreenavigationfmt{\glsnavigation}%

```

Can't use \@afterheading here as it messes with the first item of the group.

```

15755   \glstreegroupheaderskip
15756   }%
15757   \renewcommand*\{glsgroupheading}[1]{%
15758     \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15759     \glstreePreHeader{##1}{\glsxtr@grptitle}%
15760     \par
15761     \def\@gls@prevlevel{-1}%
15762     \hangindentOpt\relax
15763     \parindentOpt\relax
15764     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%

```

Can't use \@afterheading here as it messes with the first item of the group.

```

15765   \glstreegroupheaderskip
15766   }%
15767 }
15768 }%
15769 {%
15770 }

```

2.9 Multicolumn Styles

Adjust mcolindexgroup to discourage page breaks after the group headings.

```
15771 \ifdef{\@glsstyle@mcolindexgroup}
15772 {%
15773   \renewglossarystyle{mcolindexgroup}{%
15774     \setglossarystyle{mcolindex}%
15775     \renewcommand*{\glsgroupheading}[1]{%
15776       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15777       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15778       \item\glstreegroupheaderfmt{\glsxtr@grptitle}%
15779       \glstreegroupheaderskip\@afterheading
15780     }%
15781   }
15782 }%
15783 {%
15784 }
```

Similarly for mcolindexhypergroup.

```
15785 \ifdef{\@glsstyle@mcolindexhypergroup}
15786 {%
15787   \renewglossarystyle{mcolindexhypergroup}{%
15788     \setglossarystyle{mcolindex}%
15789     \renewcommand*{\glossaryheader}{%
15790       \item\glstreenavigationfmt{\glsnavigation}%
15791       \glstreegroupheaderskip\@afterheading
15792     }%
15793     \renewcommand*{\glsgroupheading}[1]{%
15794       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15795       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15796       \item\glstreegroupheaderfmt
15797         {\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15798       \glstreegroupheaderskip\@afterheading
15799     }%
15800   }
15801 }%
15802 {%
15803 }
```

Similarly for mcolindexspannav.

```
15804 \ifdef{\@glsstyle@mcolindexspannav}
15805 {%
15806   \renewglossarystyle{mcolindexspannav}{%
15807     \setglossarystyle{index}%
15808     \renewenvironment{theglossary}%
15809     {%
15810       \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]%
15811       \setlength{\parindent}{0pt}%
15812       \setlength{\parskip}{0pt plus 0.3pt}%

```



```

15813     \let\item\glstreeitem}%
15814 {\end{multicols}}}%
15815 \renewcommand*\glsgroupheading}[1]{%
15816     \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
15817     \glstreePreHeader{##1}{\glxtr@grptitle}%
15818     \item\glstreegroupheaderfmt
15819         {\glsnavigationhypertarget{##1}{\glxtr@grptitle}}}%
15820     \glstreegroupheaderskip\@afterheading
15821 }%
15822 }
15823 }%
15824 {%
15825 }

```

Similarly for mcoltreegroup.

```

15826 \ifdef{\@glsstyle@mcoltreegroup}
15827 {%
15828     \renewglossarystyle{mcoltreegroup}{%
15829         \setglossarystyle{mcoltree}%
15830         \renewcommand*\glsgroupheading}[1]{%
15831             \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
15832             \glstreePreHeader{##1}{\glxtr@grptitle}%
15833             \par\noindent\glstreegroupheaderfmt{\glxtr@grptitle}%
15834             \glstreegroupheaderskip\@afterheading
15835         }%
15836     }
15837 }%
15838 {%
15839 }

```

Similarly for mcoltreehypergroup.

```

15840 \ifdef{\@glsstyle@mcoltreehypergroup}
15841 {%
15842     \renewglossarystyle{mcoltreehypergroup}{%
15843         \setglossarystyle{mcoltree}%
15844         \renewcommand*\glossaryheader{%
15845             \par\noindent\glstreenavigationfmt{\glsnavigation}%
15846             \glstreegroupheaderskip
15847         }%
15848         \renewcommand*\glsgroupheading}[1]{%
15849             \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
15850             \glstreePreHeader{##1}{\glxtr@grptitle}%
15851             \par\noindent
15852             \glstreegroupheaderfmt{\glsnavigationhypertarget{##1}{\glxtr@grptitle}}}%
15853             \glstreegroupheaderskip\@afterheading
15854         }%
15855     }
15856 }%
15857 {%
15858 }

```

Similarly for mcoltreesspannav.

```

15859 \ifdef{\@glsstyle@mcoltreesspannav}
15860 {%
15861   \renewglossarystyle{mcoltreesspannav}{%
15862     \setglossarystyle{tree}%
15863     \renewenvironment{theglossary}%
15864     {%
15865       \begin{multicols}{\glsmcols}%
15866       [\noindent\glstreenavigationfmt{\glsnavigation}}%
15867       \setlength{\parindent}{0pt}%
15868       \setlength{\parskip}{0pt plus 0.3pt}%
15869     }%
15870     {\end{multicols}}}%
15871   \renewcommand*{\glsgroupheading}[1]{%
15872     \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15873     \glstreePreHeader{##1}{\glsxtr@grptitle}%
15874     \par\noindent
15875     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15876     \glstreegroupheaderskip\@afterheading
15877   }%
15878 }
15879 }%
15880 {%
15881 }

```

Similarly for mcoltreenonamegroup.

```

15882 \ifdef{\@glsstyle@mcoltreenonamegroup}
15883 {%
15884   \renewglossarystyle{mcoltreenonamegroup}{%
15885     \setglossarystyle{mcoltreenoname}%
15886     \renewcommand{\glsgroupheading}[1]{%
15887       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15888       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15889       \par\noindent\glstreegroupheaderfmt{\glsxtr@grptitle}%
15890       \glstreegroupheaderskip\@afterheading
15891     }%
15892   }
15893 }%
15894 {%
15895 }

```

Similarly for mcoltreenonamehypergroup.

```

15896 \ifdef{\@glsstyle@mcoltreenonamehypergroup}
15897 {%
15898   \renewglossarystyle{mcoltreenonamehypergroup}{%
15899     \setglossarystyle{mcoltreenoname}%
15900     \renewcommand*{\glossaryheader}{%
15901       \par\noindent\glstreenavigationfmt{\glsnavigation}%
15902       \glstreegroupheaderskip
15903     }%

```

```

15904 \renewcommand*{\glsgroupheading}[1]{%
15905     \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
15906     \glstreePreHeader{##1}{\glxtr@grptitle}%
15907     \par\noindent
15908     \glstreegroupheaderfmt{\glsnabhypertarget{##1}{\glxtr@grptitle}}}%
15909     \glstreegroupheaderskip\@afterheading}%
15910 }
15911 }%
15912 {%
15913 }

```

Similarly for mcoltreenonamespannav.

```

15914 \ifdef{\@glsstyle@mcoltreenonamespannav}
15915 {%
15916     \renewglossarystyle{mcoltreenonamespannav}{%
15917         \setglossarystyle{treenoname}%
15918         \renewenvironment{theglossary}%
15919         {%
15920             \begin{multicols}{\glsmcols}%
15921             [\noindent\glstreenavigationfmt{\glsnavigation}]%
15922             \setlength{\parindent}{0pt}%
15923             \setlength{\parskip}{0pt plus 0.3pt}%
15924         }%
15925         {\end{multicols}}}%
15926     \renewcommand*{\glsgroupheading}[1]{%
15927         \glxtrgetgrouptitle{##1}{\glxtr@grptitle}%
15928         \glstreePreHeader{##1}{\glxtr@grptitle}%
15929         \par\noindent
15930         \glstreegroupheaderfmt{\glsnabhypertarget{##1}{\glxtr@grptitle}}}%
15931         \glstreegroupheaderskip\@afterheading}%
15932     }
15933 }%
15934 {%
15935 }

```

mcolaltnav needs adjusting so that it uses \glxtraltnavInit This doesn't use \mbox{} \par which would unbalance the top of the columns.

```

15936 \ifdef{\@glsstyle@mcolaltnav}
15937 {%
15938     \renewglossarystyle{mcolaltnav}{%
15939         \setglossarystyle{altnav}%
15940         \renewenvironment{theglossary}%
15941         {%
15942             \glxtraltnavInit
15943             \def\@gls@prevlevel{-1}%
15944             \begin{multicols}{\glsmcols}%
15945         }%
15946         {\par\end{multicols}}}%
15947     }
15948 }%

```

```
15949 {%
15950 }
```

Redefine mcolalmtreegroup to discourage page breaks after the group headings.

```
15951 \ifdef{\@glsstyle@mcolalmtreegroup}
15952 {%
15953   \renewglossarystyle{mcolalmtreegroup}{%
15954     \setglossarystyle{mcolalmtree}%
15955     \renewcommand{\glsgroupheading}[1]{%
15956       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15957       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15958       \par
15959       \def\@gls@prevlevel{-1}%
15960       \hangindent0pt\relax
15961       \parindent0pt\relax
15962       \glstreegroupheaderfmt{\glsxtr@grptitle}%
15963       \glstreegroupheaderskip
15964     }%
15965   }
15966 }%
15967 {%
15968 }
```

Similarly for mcolalmtreehypergroup.

```
15969 \ifdef{\@glsstyle@mcolalmtreehypergroup}
15970 {%
15971   \renewglossarystyle{mcolalmtreehypergroup}{%
15972     \setglossarystyle{mcolalmtree}%
15973     \renewcommand*\glossaryheader{%
15974       \par
15975       \def\@gls@prevlevel{-1}%
15976       \hangindent0pt\relax
15977       \parindent0pt\relax
15978       \glstreenavigationfmt{\glsnavigation}%
15979       \glstreegroupheaderskip
15980     }%
15981     \renewcommand*\glsgroupheading[1]{%
15982       \glsxtrgetgrouptitle{##1}{\glsxtr@grptitle}%
15983       \glstreePreHeader{##1}{\glsxtr@grptitle}%
15984       \par
15985       \def\@gls@prevlevel{-1}%
15986       \hangindent0pt\relax
15987       \parindent0pt\relax
15988       \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsxtr@grptitle}}%
15989       \glstreegroupheaderskip
15990     }%
15991   }
15992 }%
15993 {%
15994 }
```

Similarly for mcolalttreespannav.

```
15995 \ifdef{\@glsstyle@mcolalttreespannav}
15996 {%
15997   \renewglossarystyle{mcolalttreespannav}{%
15998     \setglossarystyle{alttree}%
15999     \renewenvironment{theglossary}%
16000     {%
16001       \glstralttreeInit
16002       \def\@gls@prevlevel{-1}%
16003       \begin{multicols}{\glsmcols}%
16004         [\noindent\glstreenavigationfmt{\glsnavigation}]]%
16005     }%
16006     {\par\end{multicols}}}%
16007   \renewcommand*\@gls@groupheading}[1]{%
16008     \glstrgetgrouptitle{##1}{\glstr@grptitle}%
16009     \glstreePreHeader{##1}{\glstr@grptitle}%
16010     \par
16011     \def\@gls@prevlevel{-1}%
16012     \hangindentOpt\relax
16013     \parindentOpt\relax
16014     \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glstr@grptitle}}}%
16015     \glstreegroupheaderskip
16016   }%
16017 }
16018 }%
16019 {%
16020 }
```

Reset the default style

```
16021 \ifx\@glossary@default@style\relax
16022 \else
16023   \setglossarystyle{\@glstr@current@style}
16024 \fi
```

3 bookindex style (glossary-bookindex.sty)

3.1 Package Initialisation and Options

```
16025 \NeedsTeXFormat{LaTeX2e}
16026 \ProvidesPackage{glossary-bookindex}[2020/02/28 v1.43 (NLCT)]
```

Load required packages.

```
16027 \RequirePackage{multicol}
16028 \RequirePackage{glossary-tree}
```

trbookindexcols Number of columns.

```
16029 \newcommand{\glstrbookindexcols}{2}
```

trbookindexname Format used for top-level entries. (Argument is the label.)

```
16030 \newcommand*{\glstrbookindexname}[1]{\glossentryname{#1}}
```

ookindexsubname Format used for sub entries.

```
16031 \newcommand*{\glstrbookindexsubname}[1]{\glstrbookindexname{#1}}
```

sxtrprelocation Provide in case glossaries-stylemods isn't loaded.

```
16032 \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.

```
16033 \newcommand*{\glsxtrbookindexprelocation}[1]{%
16034   \glxtrifhasfield{location}{#1}%
16035   {,\glsxtrprelocation}%
16036   {\glsxtrprelocation}%
16037 }
```

xsubprelocation Separator used before location list for sub-entries.

```
16038 \newcommand*{\glsxtrbookindexsubprelocation}[1]{%
16039   \glsxtrbookindexprelocation{#1}%
16040 }
```

okindexlocation

```
\glsxtrbookindexlocation{\label}{\location}
```

Displays the location.

```
16041 \newcommand*{\glsxtrbookindexlocation}[2]{#2}
```

indexsublocation

```
\glxtrbookindexlocation{<label>}{<location>}
```

Displays the location for sub-entries.

```
16042 \newcommand*{\glxtrbookindexsublocation}{\glxtrbookindexlocation}
```

parentchildsep Separator used between top-level parent and child entry.

```
16043 \newcommand{\glxtrbookindexparentchildsep}{\nopagebreak}
```

parentsubchildsep Separator used between sub-level parent and child entry.

```
16044 \newcommand{\glxtrbookindexparentsubchildsep}{\glxtrbookindexparentchildsep}
```

bookindexbetween Between two top-level entries identified by the labels in the arguments.

```
16045 \newcommand{\glxtrbookindexbetween}[2]{}
```

indexsubbetween Between two level 1 entries identified by the labels in the arguments.

```
16046 \newcommand{\glxtrbookindexsubbetween}[2]{}
```

exsubsubbetween Between two level 2 entries identified by the labels in the arguments.

```
16047 \newcommand{\glxtrbookindexsubsubbetween}[2]{}
```

indexatendgroup At the end of a letter group. The argument is the index of the last top-level entry.

```
16048 \newcommand{\glxtrbookindexatendgroup}[1]{}
```

exsubatendgroup At the end of a letter group. The argument is the index of the last level 1 entry.

```
16049 \newcommand{\glxtrbookindexsubatendgroup}[1]{}
```

subsubatendgroup At the end of a letter group. The argument is the index of the last level 2 entry.

```
16050 \newcommand{\glxtrbookindexsubsubatendgroup}[1]{}
```

bookindexgroupskip Group separator.

```
16051 \newcommand{\glxtrbookindexgroupskip}{\ifglxnogroupskip\else\indexspace\fi}
```

Format group title.

indexformatheader Group separator.

```
16052 \newcommand*{\glxtrbookindexformatheader}[1]{%
```

```
16053 \par{\centering\glstreegroupheaderfmt{#1}\par}%
```

```
16054 }
```

bookindexbookmark Book mark group heading if supported.

```
16055 \ifdef\pdfbookmark
```

```
16056 {%
```

```
16057 \newcommand*{\glxtrbookindexbookmark}[2]{%
```

```
16058 \ifdefstring{@@glossarysec}{chapter}%
```

```
16059 {\pdfbookmark[1]{#1}{#2}}%
```

```

16060     {\pdfbookmark[2]{#1}{#2}}%
16061   }
16062 }
16063 {%
16064   \newcommand*{\glxtrbookindexbookmark}[2]{%
16065 }

```

bookmarkprefix Make the bookmark label prefix used for letter groups depend on the glossary label (instead of original hardcoded “index.”).

```

16066 \newcommand*{\glxtrbookindexbookmarkprefix}{\currentglossary.}

```

indexcolspread

```

16067 \newcommand*{\glxtrbookindexcolspread}{}

```

indexmulticolseenv

```

16068 \newcommand*{\glxtrbookindexmulticolseenv}{multicols}

```

Define the style.

```

16069 \newglossarystyle{bookindex}{%
16070   \setglossarystyle{index}%
16071   \renewenvironment{theglossary}%
16072   {%
16073     \ifnum\glxtrbookindexcols>1\relax
16074     \ifdefempty\glxtrbookindexcolspread
16075     {%
16076       \edef\glxtr@beginbookindex{%
16077         \noexpand\begin{\glxtrbookindexmulticolseenv}
16078         {\glxtrbookindexcols}%
16079       }%
16080     }%
16081     {%
16082       \edef\glxtr@beginbookindex{%
16083         \noexpand\begin{\glxtrbookindexmulticolseenv}%
16084         {\glxtrbookindexcols}{\glxtrbookindexcolspread}%
16085       }%
16086     }%
16087   \else
16088     \def\glxtr@beginbookindex{}%
16089   \fi
16090   \glxtr@beginbookindex
16091   \setlength{\parindent}{0pt}%
16092   \setlength{\parskip}{0pt plus 0.3pt}%
16093   \let\@glxtr@bookindex@sep\glxtrbookindexparentchildsep
16094   \let\@glxtr@bookindex@subsep\glxtrbookindexparentsubchildsep
16095   \let\@glxtr@bookindex@between\@gobble
16096   \let\@glxtr@bookindex@subbetween\@gobble
16097   \let\@glxtr@bookindex@subsubbetween\@gobble
16098   \let\@glxtr@bookindex@atendgroup\relax
16099   \let\@glxtr@bookindex@subatendgroup\relax

```



```

16100 \let\@glxstr@bookindex@subsubatendgroup\relax
16101 \let\@glxstr@bookindexgroupskip\relax
16102 }%
16103 {%

```

Do end group hooks.

```

16104 \@glxstr@bookindex@subsubatendgroup
16105 \@glxstr@bookindex@subatendgroup
16106 \@glxstr@bookindex@atendgroup

```

End multicol environment.

```

16107 \ifnum\glxstrbookindexcols>1\relax
16108 \edef\glxstr@endbookindex{%
16109 \noexpand\end{\glxstrbookindexmulticolscenv}%
16110 }%
16111 \else
16112 \def\glxstr@endbookindex{%
16113 \fi
16114 \glxstr@endbookindex
16115 }%

```

Use ragged right as columns are likely to be narrow and indexes tend not to be fully justified.

```

16116 \renewcommand*{\glossaryheader}{\raggedright}%

```

Top level entry format.

```

16117 \renewcommand*{\glossentry}[2]{%

```

Do separator.

```

16118 \@glxstr@bookindex@between{##1}%

```

Update separators.

```

16119 \let\@glxstr@bookindex@sep\glxstrbookindexparentchildsep
16120 \let\@glxstr@bookindex@subsep\glxstrbookindexparentsubchildsep
16121 \let\@glxstr@bookindex@subbetween\@gobble
16122 \let\@glxstr@bookindex@subsubbetween\@gobble
16123 \edef\@glxstr@bookindex@between{%
16124 \noexpand\glxstrbookindexbetween{##1}%
16125 }%
16126 \edef\@glxstr@bookindex@atendgroup{%
16127 \noexpand\glxstrbookindexatendgroup{##1}%
16128 }%
16129 \let\@glxstr@bookindex@subatendgroup\relax
16130 \let\@glxstr@bookindex@subsubatendgroup\relax

```

Format entry.

```

16131 \glstreeitem
16132 \glstentryitem{##1}%
16133 \glstarget{##1}{\glxstrbookindexname{##1}}%
16134 \glxstrbookindexprelocation{##1}%
16135 \glxstrbookindexlocation{##1}{##2}%
16136 }%
16137 \renewcommand{\subglossentry}[3]{%
16138 \ifcase##1\relax

```

Level 0 (shouldn't happen as that's formatted with \glossentry).

```
16139      \glstreeitem
16140      \or
```

Level 1.

```
16141      \@glxtr@bookindex@sep
16142      \@glxtr@bookindex@subbetween{##2}%
16143      \let\@glxtr@bookindex@sep\relax
```

Update separators.

```
16144      \let\@glxtr@bookindex@subsubbetween\@gobble
16145      \let\@glxtr@bookindex@subsep\glxtrbookindexparentsubchildsep
16146      \edef\@glxtr@bookindex@subbetween{%
16147          \noexpand\glxtrbookindexsubbetween{##2}%
16148      }%
16149      \edef\@glxtr@bookindex@atsubendgroup{%
16150          \noexpand\glxtrbookindexatsubendgroup{##1}%
16151      }%
```

Start sub-item.

```
16152      \glstreesubitem
16153      \glssubentryitem{##2}%
16154      \else
```

All other levels.

```
16155      \@glxtr@bookindex@subsep
16156      \@glxtr@bookindex@subsubbetween{##2}%
```

Update separators.

```
16157      \let\@glxtr@bookindex@subsep\relax
16158      \edef\@glxtr@bookindex@subsubbetween{%
16159          \noexpand\glxtrbookindexsubsubbetween{##2}%
16160      }%
16161      \edef\@glxtr@bookindex@atsubsubendgroup{%
16162          \noexpand\glxtrbookindexatsubsubendgroup{##1}%
16163      }%
```

Start sub-sub-item.

```
16164      \glstreesubsubitem
16165      \fi
```

Format entry.

```
16166      \glstarget{##2}{\glxtrbookindexsubname{##2}}%
16167      \glxtrbookindexsubprelocation{##2}%
16168      \glxtrbookindexsublocation{##2}{##3}%
16169      }%
```

The group skip is moved to the group heading to avoid interfering with the end letter group hooks.

```
16170      \renewcommand*{\glsgroupskip}{}%
```

Group heading format.

```
16171      \renewcommand*{\glsgroupheading}[1]{%
```

Do end group hooks.

```
16172 \glxtr@bookindex@subsubatendgroup
16173 \glxtr@bookindex@subatendgroup
16174 \glxtr@bookindex@atendgroup
16175 \glxtr@bookindexgroupskip
```

Update separators.

```
16176 \let\glxtr@bookindexgroupskip\glxtrbookindexgroupskip
16177 \let\glxtr@bookindex@between\@gobble
16178 \let\glxtr@bookindex@atendgroup\relax
16179 \let\glxtr@bookindex@subatendgroup\relax
16180 \let\glxtr@bookindex@subsubatendgroup\relax
```

Fetch the group title from the label supplied in #1.

```
16181 \glxtrgetgrouptitle{##1}{\glxtrcurrentgrptitle}%
```

Do the PDF bookmark if supported.

```
16182 \glxtrbookindexbookmark{\glxtrcurrentgrptitle}{\glxtrbookindexbookmarkprefix##1}%
```

Format the group title.

```
16183 \glxtrbookindexformatheader{\glxtrcurrentgrptitle}%
16184 \nopagebreak\indexspace\nopagebreak\@afterheading
16185 }%
16186 }
```

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.

`bookindexthepage` The \@printglossary sets \currentglossary to the current glossary label. This is used as a prefix in case the page number is reset.

```
16187 \newcommand{\glxtrbookindexthepage}{%
16188 \ifdef\currentglossary{\currentglossary.\arabic{page}}{\arabic{page}}%
16189 }
```

`bookindexmarkentry` Writes entry information to the .aux file. The argument is the entry label.

```
16190 \newcommand*{\glxtrbookindexmarkentry}[1]{%
16191 \protected@write\@auxout
16192 {\let\glxtrbookindexthepage\relax}%
16193 {\string\glxtr@setbookindexmark{\glxtrbookindexthepage}{#1}}%
16194 }
```

`setbookindexmark`

```
16195 \newcommand*{\glxtr@setbookindexmark}[2]{%
16196 \ifcsundef\glxtr@idxfirstmark@#1}%
16197 {\csgdef\glxtr@idxfirstmark@#1}{#2}}%
16198 {}%
16199 \csgdef\glxtr@idxlastmark@#1}{#2}%
16200 }
```

dexfirstmarkfmt

```
16201 \newcommand*{\glxtrbookindexfirstmarkfmt}[1]{%
16202   \glstryname{#1}%
16203 }
```

kindexfirstmark

```
16204 \newcommand*{\glxtrbookindexfirstmark}{%
16205   \letcs{\glxtr@label}{\glxtr@idxfirstmark@\glxtrbookindexthepage}%
16206   \ifdef\glxtr@label
16207     {\glxtrbookindexfirstmarkfmt{\glxtr@label}}%
16208     {}%
16209 }
```

ndexlastmarkfmt

```
16210 \newcommand*{\glxtrbookindexlastmarkfmt}[1]{%
16211   \glstryname{#1}%
16212 }
```

okindexlastmark

```
16213 \newcommand*{\glxtrbookindexlastmark}{%
16214   \letcs{\glxtr@label}{\glxtr@idxlastmark@\glxtrbookindexthepage}%
16215   \ifdef\glxtr@label
16216     {\glxtrbookindexlastmarkfmt{\glxtr@label}}%
16217     {}%
16218 }
```

4 longextra styles (glossary-longextra.sty)

4.1 Package Initialisation and Options

Provides additional long styles.

```
16219 \NeedsTeXFormat{LaTeX2e}  
16220 \ProvidesPackage{glossary-longextra}[2020/02/28 v1.43 (NLCT)]
```

Load required packages.

```
16221 \RequirePackage{glossary-longbooktabs}
```

ongextraNameFmt

```
\glslongextraNameFmt{<label>}
```

Governs the way the name is displayed.

```
16222 \newcommand{\glslongextraNameFmt}[1]{%  
16223   \glsentryitem{#1}\glstarget{#1}{\glossentryname{#1}}%  
16224 }
```

ongextraDescFmt

```
\glslongextraDescFmt{<label>}
```

Governs the way the description is displayed.

```
16225 \newcommand{\glslongextraDescFmt}[1]{%  
16226   \glossentrydesc{#1}\glspostdescription  
16227 }
```

gextraSymbolFmt

```
\glslongextraSymbolFmt{<label>}
```

Governs the way the symbol is displayed.

```
16228 \newcommand{\glslongextraSymbolFmt}[1]{\glossentrysymbol{#1}}
```

extraLocationFmt

```
\glslongextraLocationFmt{<label>}{<location list>}
```

Governs the way the location is displayed.

```
16229 \newcommand{\glslongextraLocationFmt}[2]{#2}
```

extraSubNameFmt

```
\glslongextraSubNameFmt{<level>}{<label>}
```

Governs the way the child name is displayed. Just does the sub-entry counter, if enabled, and the target.

```
16230 \newcommand{\glslongextraSubNameFmt}[2]{%
16231   \glssubentryitem{#2}\glstarget{#2}{\strut}%
16232 }
```

extraSubDescFmt

```
\glslongextraSubDescFmt{<level>}{<label>}
```

Governs the way the child description is displayed.

```
16233 \newcommand{\glslongextraSubDescFmt}[2]{%
16234   \glslongextraDescFmt{#2}%
16235 }
```

extraSubSymbolFmt

```
\glslongextraSubSymbolFmt{<level>}{<label>}
```

Governs the way the child symbol is displayed.

```
16236 \newcommand{\glslongextraSubSymbolFmt}[2]{%
16237   \glslongextraSymbolFmt{#2}%
16238 }
```

extraSubLocationFmt

```
\glslongextraSubLocationFmt{<level>}{<label>}{<location list>}
```

Governs the way the child location list is displayed.

```
16239 \newcommand{\glslongextraSubLocationFmt}[3]{#3}
```

`gextraNameAlign` Alignment for the name column.
16240 `\newcommand{\glslongextraNameAlign}{l}`

`gextraDescAlign` Alignment for the description column.
16241 `\newcommand{\glslongextraDescAlign}{>{\raggedright}p{\glstdescwidth}}`

`extraSymbolAlign` Alignment for the symbol column.
16242 `\newcommand{\glslongextraSymbolAlign}{c}`

`raLocationAlign` Alignment for the location column.
16243 `\newcommand{\glslongextraLocationAlign}{>{\raggedright}p{\glspagelistwidth}}`

`extraGroupHeading` Used to format the letter group headings. The first argument is the number of columns in the table. The second is the group *label* (not the title).
16244 `\newcommand{\glslongextraGroupHeading}[2]{}`

`extraHeaderFormat` Format for the column headers.
16245 `\newcommand{\glslongextraHeaderFmt}[1]{\textbf{#1}}`

`aNameDescHeader`
16246 `\newcommand{\glslongextraNameDescHeader}{%`
16247 `\glslongextraNameDescTabularHeader\endhead`
16248 `\glslongextraNameDescTabularFooter\endfoot`
16249 `}`

`scTabularHeader`
16250 `\newcommand{\glslongextraNameDescTabularHeader}{%`
16251 `\toprule`
16252 `\glslongextraHeaderFmt\entryname &`
16253 `\glslongextraHeaderFmt\descriptionname\tabularnewline`
16254 `\midrule`
16255 `}`

`scTabularFooter`
16256 `\newcommand{\glslongextraNameDescTabularFooter}{%`
16257 `\bottomrule`
16258 `}`

Unlike the `alttree` style, there aren't different widths for the hierarchical levels.

`gextraSetWidest` Provide in case the tree styles haven't been loaded.
16259 `\newcommand*{\glslongextraSetWidest}[1]{%`
16260 `\def\@glslongextrawidestname{#1}%`
16261 `}`

`extrawidestname` Pick up the widest name from the `alttree` style if it has been set. (Will expand to nothing otherwise.)
16262 `\newcommand*{\@glslongextrawidestname}{\csuse{@glswidestname}}`

traUpdateWidest

```

16263 \newcommand*{\glslongextraUpdateWidest}[1]{%
16264   \ifundef\@glslongextrawidestname
16265   {\def\@glslongextrawidestname{#1}}%
16266   {%
16267     \settowidth{\dimen@}{\@glslongextrawidestname}%
16268     \settowidth{\dimen@ii}{#1}%
16269     \ifdim\dimen@ii>\dimen@
16270     \def\@glslongextrawidestname{#1}%
16271     \fi
16272   }%
16273 }
```

dateWidestChild

`\glslongextraUpdateWidestChild{<level>}{<text>}`

Used by `\glsxtrSetWidest` in `glossaries-extra-bib2gls`. Does nothing by default, since the default action in these styles is to omit the child name. If the child name should be displayed, then this needs to be redefined to use `\glslongextraUpdateWidest`.

```

16274 \newcommand*{\glslongextraUpdateWidestChild}[2]{}
```

traSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name and description columns.

```

16275 \newcommand{\glslongextraSetDescWidth}{%
16276   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\entryname}%
16277   \settowidth{\dimen@}{\glsnamefont{\@glslongextrawidestname}}%
16278   \ifdim\dimen@>\gls@tmplen
16279     \gls@tmplen=\dimen@
16280   \fi
```

Description width is `\linewidth` less 4`\tabcolsep` less the width of the name column.

```

16281   \setlength{\glsdescwidth}{\dimexpr\linewidth-4\tabcolsep-\gls@tmplen}%
16282 }
```

SymSetDescWidth Computes the value of `\glsdescwidth` for the styles that only have name, symbol and description columns.

```

16283 \newcommand{\glslongextraSymSetDescWidth}{%
```

Work out the size for just the name and description style.

```

16284   \glslongextraSetDescWidth
```

Now work out the symbol column width. This is assuming that the column title will be the widest text in the column.

```

16285   \settowidth{\gls@tmplen}{\glslongextraHeaderFmt\symbolname}%
```


Subtract 2\tabcolsep and the symbol header width.

```

16286 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\gls@tmplen}%
16287 }

```

LocSetDescWidth Computes the value of \glsdescwidth for the styles that only have name, location and description columns.

```

16288 \newcommand{\glslongextraLocSetDescWidth}{%
  Work out the size for just the name and description style.
16289 \glslongextraSetDescWidth
  Subtract 2\tabcolsep and the location list column width.
16290 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
16291 }

```

LocSetDescWidth Computes the value of \glsdescwidth for the styles that have name, symbol, location and description columns.

```

16292 \newcommand{\glslongextraSymLocSetDescWidth}{%
  Work out the size for just the name, symbol and description style.
16293 \glslongextraSymSetDescWidth
  Subtract 2\tabcolsep and the location list column width.
16294 \setlength{\glsdescwidth}{\dimexpr\glsdescwidth-2\tabcolsep-\glspagelistwidth}%
16295 }

```

ExtraUseTabular If true use tabular instead of longtable. Obviously only intended for short glossaries that can fit into a single page.

```

16296 \newif\ifGlsLongExtraUseTabular
16297 \GlsLongExtraUseTabularfalse

```

raTabularVAlign Only used with the tabular setting.

```

16298 \newcommand*{\glslongextraTabularVAlign}{c}

```

long-name-desc Two column style with multi-lined descriptions and header. This is similar to the longragged-booktabs style.

```

16299 \newglossarystyle{long-name-desc}%
16300 {%
16301 \ifGlsLongExtraUseTabular
16302 \renewenvironment{theglossary}%
16303 {%
16304 \glslongextraSetDescWidth
16305 \edef\@glslongextra@begintab{%
16306 \noexpand\begin{tabular}{\glslongextraTabularVAlign}{%
16307 \expandonce\glslongextraNameAlign
16308 \expandonce\glslongextraDescAlign}}%
16309 \@glslongextra@begintab
16310 }%
16311 {%

```

```

16312     \glslongextraNameDescTabularFooter
16313     \end{tabular}%
16314 }%
16315 \renewcommand*{\glossaryheader}{\glslongextraNameDescTabularHeader}%
16316 \else
16317 \renewenvironment{theglossary}%
16318 {%
16319     \glspatchLToutput
16320     \glslongextraSetDescWidth
16321     \edef\@glslongextra@begintab{%
16322         \noexpand\begin{longtable}{%
16323             \expandonce\glslongextraNameAlign
16324             \expandonce\glslongextraDescAlign}}%
16325     \@glslongextra@begintab
16326 }%
16327 {\end{longtable}}%
16328 \renewcommand*{\glossaryheader}{\glslongextraNameDescHeader}%
16329 \fi
16330 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
16331 \renewcommand{\glossentry}[2]{%
16332     \glslongextraNameFmt{##1} &
16333     \glslongextraDescFmt{##1}\tabularnewline
16334 }%
16335 \renewcommand{\subglossentry}[3]{%
16336     \glslongextraSubNameFmt{##1}{##2}
16337     &
16338     \glslongextraSubDescFmt{##1}{##2}%
16339     \tabularnewline
16340 }%
16341 \ifglsgroupskip
16342     \renewcommand*{\glsgroupskip}{}%
16343 \else
16344     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16345 \fi
16346 }

```

cLocationHeader

```

16347 \newcommand{\glslongextraNameDescLocationHeader}{%
16348     \glslongextraNameDescLocationTabularHeader\endhead
16349     \glslongextraNameDescLocationTabularFooter\endfoot
16350 }

```

onTabularHeader

```

16351 \newcommand{\glslongextraNameDescLocationTabularHeader}{%
16352     \toprule
16353     \glslongextraHeaderFmt\entryname &
16354     \glslongextraHeaderFmt\descriptionname &
16355     \glslongextraHeaderFmt\pagelistname\tabularnewline
16356     \midrule

```

16357 }

onTabularFooter

```
16358 \newcommand{\glslongextraNameDescLocationTabularFooter}{%
16359   \bottomrule
16360 }
```

g-name-desc-loc Three columns: name, description and location list.

```
16361 \newglossarystyle{long-name-desc-loc}%
16362 {%
16363   \ifGlsLongExtraUseTabular
16364     \renewenvironment{theglossary}%
16365       {%
16366         \glslongextraLocSetDescWidth
16367         \edef\@glslongextra@begintab{%
16368           \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16369             \expandonce\glslongextraNameAlign
16370             \expandonce\glslongextraDescAlign
16371             \expandonce\glslongextraLocationAlign
16372           }}%
16373         \@glslongextra@begintab
16374       }%
16375       {%
16376         \glslongextraNameDescLocationTabularFooter
16377         \end{tabular}%
16378       }%
16379     \renewcommand*\glossaryheader{\glslongextraNameDescLocationTabularHeader}%
16380   \else
16381     \renewenvironment{theglossary}%
16382       {%
16383         \glspatchLToutput
16384         \glslongextraLocSetDescWidth
16385         \edef\@glslongextra@begintab{%
16386           \noexpand\begin{longtable}%
16387             \expandonce\glslongextraNameAlign
16388             \expandonce\glslongextraDescAlign
16389             \expandonce\glslongextraLocationAlign
16390           }}%
16391         \@glslongextra@begintab
16392       }%
16393       {\end{longtable}}%
16394     \renewcommand*\glossaryheader{\glslongextraNameDescLocationHeader}%
16395   \fi
16396   \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
16397   \renewcommand{\glossentry}[2]{%
16398     \glslongextraNameFmt{##1} &
16399     \glslongextraDescFmt{##1} &
16400     \glslongextraLocationFmt{##1}{##2}\tabularnewline
16401   }%
```

```

16402 \renewcommand{\subglossentry}[3]{%
16403     \glslongextraSubNameFmt{##1}{##2}&
16404     \glslongextraSubDescFmt{##1}{##2}&
16405     \glslongextraSubLocationFmt{##1}{##2}{##3}%
16406     \tabularnewline
16407 }%
16408 \ifglsgroupskip
16409 \renewcommand*{\glsgroupskip}{}%
16410 \else
16411 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16412 \fi
16413 }

```

aDescNameHeader

```

16414 \newcommand{\glslongextraDescNameHeader}{%
16415 \glslongextraDescNameTabularHeader\endhead
16416 \glslongextraDescNameTabularFooter\endfoot
16417 }

```

meTabularHeader

```

16418 \newcommand{\glslongextraDescNameTabularHeader}{%
16419 \toprule
16420 \glslongextraHeaderFmt\descriptionname&
16421 \glslongextraHeaderFmt\entryname \tabularnewline
16422 \midrule
16423 }

```

meTabularFooter

```

16424 \newcommand{\glslongextraDescNameTabularFooter}{%
16425 \bottomrule
16426 }

```

long-desc-name Like name-desc but swaps the columns.

```

16427 \newglossarystyle{long-desc-name}%
16428 {%
16429 \ifGlsLongExtraUseTabular
16430 \renewenvironment{theglossary}%
16431 {%
16432     \glslongextraSetDescWidth
16433     \edef\@glslongextra@begintab{%
16434         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16435             \expandonce\glslongextraDescAlign
16436             \expandonce\glslongextraNameAlign}}%
16437     \@glslongextra@begintab
16438 }%
16439 {%
16440     \glslongextraDescNameTabularFooter
16441     \end{tabular}%
16442 }%

```

```

16443 \renewcommand*{\glossaryheader}{\glslongextraDescNameTabularHeader}%
16444 \else
16445 \renewenvironment{theglossary}%
16446 {%
16447 \glspatchLToutput
16448 \glslongextraSetDescWidth
16449 \edef\@glslongextra@begintab{%
16450 \noexpand\begin{longtable}{%
16451 \expandonce\glslongextraDescAlign
16452 \expandonce\glslongextraNameAlign}}%
16453 \@glslongextra@begintab
16454 }%
16455 {\end{longtable}}%
16456 \renewcommand*{\glossaryheader}{\glslongextraDescNameHeader}%
16457 \fi
16458 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{2}{##1}}%
16459 \renewcommand{\glossentry}[2]{%
16460 \glslongextraDescFmt{##1} &
16461 \glslongextraNameFmt{##1}\tabularnewline
16462 }%
16463 \renewcommand{\subglossentry}[3]{%
16464 \glslongextraSubDescFmt{##1}{##2} &
16465 \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16466 }%
16467 \ifglsgroupskip
16468 \renewcommand*{\glsgroupskip}{}%
16469 \else
16470 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16471 \fi
16472 }

```

nDescNameHeader

```

16473 \newcommand{\glslongextraLocationDescNameHeader}{%
16474 \glslongextraLocationDescNameTabularHeader\endhead
16475 \glslongextraLocationDescNameTabularFooter\endfoot
16476 }

```

meTabularHeader

```

16477 \newcommand{\glslongextraLocationDescNameTabularHeader}{%
16478 \toprule
16479 \glslongextraHeaderFmt\pagelistname&
16480 \glslongextraHeaderFmt\descriptionname&
16481 \glslongextraHeaderFmt\entryname \tabularnewline
16482 \midrule
16483 }

```

meTabularFooter

```

16484 \newcommand{\glslongextraLocationDescNameTabularFooter}{%
16485 \bottomrule

```

16486 }

g-loc-desc-name Three columns: location, description and name.

```
16487 \newglossarystyle{long-loc-desc-name}%
16488 {%
16489   \ifGlsLongExtraUseTabular
16490     {%
16491       \glslongextraLocSetDescWidth
16492       \edef\@glslongextra@begintab{%
16493         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16494           \expandonce\glslongextraLocationAlign
16495           \expandonce\glslongextraDescAlign
16496           \expandonce\glslongextraNameAlign}}%
16497       \@glslongextra@begintab
16498     }%
16499     {%
16500       \glslongextraLocationDescNameTabularFooter
16501       \end{tabular}%
16502     }%
16503     \renewcommand*\glossaryheader{\glslongextraLocationDescNameTabularHeader}%
16504   \else
16505     \renewenvironment{theglossary}%
16506     {%
16507       \glspatchLToutput
16508       \glslongextraLocSetDescWidth
16509       \edef\@glslongextra@begintab{%
16510         \noexpand\begin{longtable}{%
16511           \expandonce\glslongextraLocationAlign
16512           \expandonce\glslongextraDescAlign
16513           \expandonce\glslongextraNameAlign}}%
16514       \@glslongextra@begintab
16515     }%
16516     {\end{longtable}}%
16517     \renewcommand*\glossaryheader{\glslongextraLocationDescNameHeader}%
16518   \fi
16519   \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
16520   \renewcommand{\glossentry}[2]{%
16521     \glslongextraLocationFmt{##1}{##2} &
16522     \glslongextraDescFmt{##1} &
16523     \glslongextraNameFmt{##1}\tabularnewline
16524   }%
16525   \renewcommand{\subglossentry}[3]{%
16526     \glslongextraSubLocationFmt{##1}{##2}{##3} &
16527     \glslongextraSubDescFmt{##1}{##2} &
16528     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16529   }%
16530   \ifglsgnogroupskip
16531     \renewcommand*\glsgroupskip{}%
16532   \else
```

```

16533 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16534 \fi
16535 }

```

meDescSymHeader

```

16536 \newcommand{\glslongextraNameDescSymHeader}{%
16537 \glslongextraNameDescSymTabularHeader\endhead
16538 \glslongextraNameDescSymTabularFooter\endfoot
16539 }

```

ymTabularHeader

```

16540 \newcommand{\glslongextraNameDescSymTabularHeader}{%
16541 \toprule
16542 \glslongextraHeaderFmt\entryname &
16543 \glslongextraHeaderFmt\descriptionname &
16544 \glslongextraHeaderFmt\symbolname\tabularnewline
16545 \midrule
16546 }

```

ymTabularFooter

```

16547 \newcommand{\glslongextraNameDescSymTabularFooter}{%
16548 \bottomrule
16549 }

```

g-name-desc-sym Three column style with symbol in the third column.

```

16550 \newglossarystyle{long-name-desc-sym}%
16551 {%
16552 \ifGlsLongExtraUseTabular
16553 \renewenvironment{theglossary}%
16554 {%
16555 \glslongextraSymSetDescWidth
16556 \edef\@glslongextra@begintab{%
16557 \noexpand\begin{tabular}{\glslongextraTabularVAlign}{%
16558 \expandonce\glslongextraNameAlign
16559 \expandonce\glslongextraDescAlign
16560 \expandonce\glslongextraSymbolAlign
16561 }}%
16562 \@glslongextra@begintab
16563 }%
16564 {%
16565 \glslongextraNameDescSymTabularFooter
16566 \end{tabular}%
16567 }%
16568 \renewcommand*{\glossaryheader}{\glslongextraNameDescSymTabularHeader}%
16569 \else
16570 \renewenvironment{theglossary}%
16571 {%
16572 \glspatchLToutput
16573 \glslongextraSymSetDescWidth

```

```

16574 \edef\@glslongextra@begintab{%
16575 \noexpand\begin{longtable}{%
16576 \expandonce\glslongextraNameAlign
16577 \expandonce\glslongextraDescAlign
16578 \expandonce\glslongextraSymbolAlign
16579 }}%
16580 \@glslongextra@begintab
16581 }%
16582 {\end{longtable}}}%
16583 \renewcommand*{\glossaryheader}{\glslongextraNameDescSymHeader}%
16584 \fi
16585 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
16586 \renewcommand{\glossentry}[2]{%
16587 \glslongextraNameFmt{##1} &
16588 \glslongextraDescFmt{##1} &
16589 \glslongextraSymbolFmt{##1}\tabularnewline
16590 }%
16591 \renewcommand{\subglossentry}[3]{%
16592 \glslongextraSubNameFmt{##1}{##2} &
16593 \glslongextraSubDescFmt{##1}{##2} &
16594 \glslongextraSubSymbolFmt{##1}{##2}%
16595 \tabularnewline
16596 }%
16597 \ifglsnogroupskip
16598 \renewcommand*{\glsgroupskip}{}%
16599 \else
16600 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16601 \fi
16602 }

```

mLocationHeader

```

16603 \newcommand{\glslongextraNameDescSymLocationHeader}{%
16604 \glslongextraNameDescSymLocationTabularHeader\endhead
16605 \glslongextraNameDescSymLocationTabularFooter\endfoot
16606 }

```

onTabularHeader

```

16607 \newcommand{\glslongextraNameDescSymLocationTabularHeader}{%
16608 \toprule
16609 \glslongextraHeaderFmt\entryname &
16610 \glslongextraHeaderFmt\descriptionname &
16611 \glslongextraHeaderFmt\symbolname &
16612 \glslongextraHeaderFmt\pagelistname\tabularnewline
16613 \midrule
16614 }

```

onTabularFooter

```

16615 \newcommand{\glslongextraNameDescSymLocationTabularFooter}{%
16616 \bottomrule

```


16617 }

me-desc-sym-loc Four columns: name, description and location

```
16618 \newglossarystyle{long-name-desc-sym-loc}%
16619 {%
16620   \ifGlsLongExtraUseTabular
16621     \renewenvironment{theglossary}%
16622       {%
16623         \glslongextraSymLocSetDescWidth
16624         \edef\@glslongextra@begintab{%
16625           \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16626             \expandonce\glslongextraNameAlign
16627             \expandonce\glslongextraDescAlign
16628             \expandonce\glslongextraSymbolAlign
16629             \expandonce\glslongextraLocationAlign
16630           }}%
16631         \@glslongextra@begintab
16632       }%
16633       {%
16634         \glslongextraNameDescSymLocationTabularFooter
16635         \end{tabular}%
16636       }%
16637     \renewcommand*\glossaryheader{\glslongextraNameDescSymLocationTabularHeader}%
16638   \else
16639     \renewenvironment{theglossary}%
16640       {%
16641         \glspatchLToutput
16642         \glslongextraSymLocSetDescWidth
16643         \edef\@glslongextra@begintab{%
16644           \noexpand\begin{longtable}{%
16645             \expandonce\glslongextraNameAlign
16646             \expandonce\glslongextraDescAlign
16647             \expandonce\glslongextraSymbolAlign
16648             \expandonce\glslongextraLocationAlign
16649           }}%
16650         \@glslongextra@begintab
16651       }%
16652       {\end{longtable}}%
16653     \renewcommand*\glossaryheader{\glslongextraNameDescSymLocationHeader}%
16654   \fi
16655   \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
16656   \renewcommand{\glossentry}[2]{%
16657     \glslongextraNameFmt{##1} &
16658     \glslongextraDescFmt{##1} &
16659     \glslongextraSymbolFmt{##1}&
16660     \glslongextraLocationFmt{##1}{##2}\tabularnewline
16661   }%
16662   \renewcommand{\subglossentry}[3]{%
16663     \glslongextraSubNameFmt{##1}{##2} &
```

```

16664 \glslongextraSubDescFmt{##1}{##2} &
16665 \glslongextraSubSymbolFmt{##1}{##2}&
16666 \glslongextraSubLocationFmt{##1}{##2}{##3}%
16667 \tabularnewline
16668 }%
16669 \ifglsgroupskip
16670 \renewcommand*{\glsgroupskip}{}%
16671 \else
16672 \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16673 \fi
16674 }

```

meSymDescHeader

```

16675 \newcommand{\glslongextraNameSymDescHeader}{%
16676 \glslongextraNameSymDescTabularHeader\endhead
16677 \glslongextraNameSymDescTabularFooter\endfoot
16678 }

```

scTabularHeader

```

16679 \newcommand{\glslongextraNameSymDescTabularHeader}{%
16680 \toprule
16681 \glslongextraHeaderFmt\entryname &
16682 \glslongextraHeaderFmt\symbolname &
16683 \glslongextraHeaderFmt\descriptionname\tabularnewline
16684 \midrule
16685 }

```

scTabularFooter

```

16686 \newcommand{\glslongextraNameSymDescTabularFooter}{%
16687 \bottomrule
16688 }

```

g-name-sym-desc Three column style with symbol in the second column.

```

16689 \newglossarystyle{long-name-sym-desc}%
16690 {%
16691 \ifGlsLongExtraUseTabular
16692 \renewenvironment{theglossary}%
16693 {%
16694 \glslongextraSymSetDescWidth
16695 \edef\@glslongextra@begintab{%
16696 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16697 \expandonce\glslongextraNameAlign
16698 \expandonce\glslongextraSymbolAlign
16699 \expandonce\glslongextraDescAlign
16700 }}%
16701 \@glslongextra@begintab
16702 }%
16703 {%
16704 \glslongextraNameSymDescTabularFooter

```

```

16705     \end{tabular}%
16706   }%
16707   \renewcommand*{\glossaryheader}{\glslongextraNameSymDescTabularHeader}%
16708 \else
16709   \renewenvironment{theglossary}%
16710   {%
16711     \glspatchLToutput
16712     \glslongextraSymSetDescWidth
16713     \edef\@glslongextra@begintab{%
16714       \noexpand\begin{longtable}{%
16715         \expandonce\glslongextraNameAlign
16716         \expandonce\glslongextraSymbolAlign
16717         \expandonce\glslongextraDescAlign
16718       }}%
16719     \@glslongextra@begintab
16720   }%
16721   {\end{longtable}}%
16722   \renewcommand*{\glossaryheader}{\glslongextraNameSymDescHeader}%
16723 \fi
16724 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
16725 \renewcommand{\glossentry}[2]{%
16726   \glslongextraNameFmt{##1} &
16727   \glslongextraSymbolFmt{##1} &
16728   \glslongextraDescFmt{##1}\tabularnewline
16729 }%
16730 \renewcommand{\subglossentry}[3]{%
16731   \glslongextraSubNameFmt{##1}{##2} &
16732   \glslongextraSubSymbolFmt{##1}{##2} &
16733   \glslongextraSubDescFmt{##1}{##2}\tabularnewline
16734 }%
16735 \ifglsnogroupskip
16736   \renewcommand*{\glsgroupskip}{}%
16737 \else
16738   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16739 \fi
16740 }

```

cLocationHeader

```

16741 \newcommand{\glslongextraNameSymDescLocationHeader}{%
16742   \glslongextraNameSymDescLocationTabularHeader\endhead
16743   \glslongextraNameSymDescLocationTabularFooter\endfoot
16744 }

```

onTabularHeader

```

16745 \newcommand{\glslongextraNameSymDescLocationTabularHeader}{%
16746   \toprule
16747   \glslongextraHeaderFmt\entryname &
16748   \glslongextraHeaderFmt\symbolname &
16749   \glslongextraHeaderFmt\descriptionname &

```

```

16750 \glslongextraHeaderFmt\pagelistname\tabularnewline
16751 \midrule
16752 }

```

onTabularFooter

```

16753 \newcommand{\glslongextraNameSymDescLocationTabularFooter}{%
16754 \bottomrule
16755 }

```

me-sym-desc-loc Four column style with symbol in the second column.

```

16756 \newglossarystyle{long-name-sym-desc-loc}%
16757 {%
16758 \ifGlsLongExtraUseTabular
16759 \renewenvironment{theglossary}%
16760 {%
16761 \glslongextraSymLocSetDescWidth
16762 \edef\@glslongextra@begintab{%
16763 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16764 \expandonce\glslongextraNameAlign
16765 \expandonce\glslongextraSymbolAlign
16766 \expandonce\glslongextraDescAlign
16767 \expandonce\glslongextraLocationAlign
16768 }}%
16769 \@glslongextra@begintab
16770 }%
16771 {%
16772 \glslongextraNameSymDescLocationTabularFooter
16773 \end{tabular}}%
16774 }%
16775 \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationTabularHeader}%
16776 \else
16777 \renewenvironment{theglossary}%
16778 {%
16779 \glspatchLToutput
16780 \glslongextraSymLocSetDescWidth
16781 \edef\@glslongextra@begintab{%
16782 \noexpand\begin{longtable}{%
16783 \expandonce\glslongextraNameAlign
16784 \expandonce\glslongextraSymbolAlign
16785 \expandonce\glslongextraDescAlign
16786 \expandonce\glslongextraLocationAlign
16787 }}%
16788 \@glslongextra@begintab
16789 }%
16790 {\end{longtable}}%
16791 \renewcommand*\glossaryheader{\glslongextraNameSymDescLocationHeader}%
16792 \fi
16793 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{4}{##1}}%
16794 \renewcommand{\glossentry}[2]{%

```

```

16795 \glslongextraNameFmt{##1} &
16796 \glslongextraSymbolFmt{##1} &
16797 \glslongextraDescFmt{##1} &
16798 \glslongextraLocationFmt{##1}{##2}\tabularnewline
16799 }%
16800 \renewcommand{\subglossentry}[3]{%
16801 \glslongextraSubNameFmt{##1}{##2} &
16802 \glslongextraSubSymbolFmt{##1}{##2} &
16803 \glslongextraSubDescFmt{##1}{##2} &
16804 \glslongextraSubLocationFmt{##1}{##2}{##3}\tabularnewline
16805 }%
16806 \ifglsgnognogroupskip
16807 \renewcommand*\{\glsgroupskip}{}%
16808 \else
16809 \renewcommand*\{\glsgroupskip}{\glspenaltygroupskip}%
16810 \fi
16811 }

```

mDescNameHeader

```

16812 \newcommand{\glslongextraSymDescNameHeader}{%
16813 \glslongextraSymDescNameTabularHeader\endhead
16814 \glslongextraSymDescNameTabularFooter\endfoot
16815 }

```

meTabularHeader

```

16816 \newcommand{\glslongextraSymDescNameTabularHeader}{%
16817 \toprule
16818 \glslongextraHeaderFmt\symbolname &
16819 \glslongextraHeaderFmt\descriptionname &
16820 \glslongextraHeaderFmt\entryname\tabularnewline
16821 \midrule
16822 }

```

meTabularFooter

```

16823 \newcommand{\glslongextraSymDescNameTabularFooter}{%
16824 \bottomrule
16825 }

```

g-sym-desc-name Three column style with symbol in the first column, description in the second and name in the third.

```

16826 \newglossarystyle{long-sym-desc-name}%
16827 {%
16828 \ifGlsLongExtraUseTabular
16829 \renewenvironment{theglossary}%
16830 {%
16831 \glslongextraSymSetDescWidth
16832 \edef\@glslongextra@begintab{%
16833 \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16834 \expandonce\glslongextraSymbolAlign

```

```

16835         \expandonce\glslongextraDescAlign
16836         \expandonce\glslongextraNameAlign
16837     } }%
16838     \@glslongextra@begintab
16839 }%
16840 {%
16841     \glslongextraSymDescNameTabularFooter
16842     \end{tabular}%
16843 }%
16844 \renewcommand*{\glossaryheader}{\glslongextraSymDescNameTabularHeader}%
16845 \else
16846 \renewenvironment{theglossary}%
16847 {%
16848     \glspatchLToutput
16849     \glslongextraSymSetDescWidth
16850     \edef\@glslongextra@begintab{%
16851         \noexpand\begin{longtable}{%
16852             \expandonce\glslongextraSymbolAlign
16853             \expandonce\glslongextraDescAlign
16854             \expandonce\glslongextraNameAlign
16855         } }%
16856     \@glslongextra@begintab
16857 }%
16858     {\end{longtable}}%
16859 \renewcommand*{\glossaryheader}{\glslongextraSymDescNameHeader}%
16860 \fi
16861 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{3}{##1}}%
16862 \renewcommand{\glossentry}[2]{%
16863     \glslongextraSymbolFmt{##1} &
16864     \glslongextraDescFmt{##1} &
16865     \glslongextraNameFmt{##1}\tabularnewline
16866 }%
16867 \renewcommand{\subglossentry}[3]{%
16868     \glslongextraSubSymbolFmt{##1}{##2} &
16869     \glslongextraSubDescFmt{##1}{##2} &
16870     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16871 }%
16872 \ifglsgroupskip
16873     \renewcommand*{\glsgroupskip}{}%
16874 \else
16875     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16876 \fi
16877 }

```

mDescNameHeader

```

16878 \newcommand{\glslongextraLocationSymDescNameHeader}{%
16879 \glslongextraLocationSymDescNameTabularHeader\endhead
16880 \glslongextraLocationSymDescNameTabularFooter\endfoot
16881 }

```

meTabularHeader

```

16882 \newcommand{\glslongextraLocationSymDescNameTabularHeader}{%
16883   \toprule
16884   \glslongextraHeaderFmt\pagelistname &
16885   \glslongextraHeaderFmt\symbolname &
16886   \glslongextraHeaderFmt\descriptionname &
16887   \glslongextraHeaderFmt\entryname\tabularnewline
16888   \midrule
16889 }

```

meTabularFooter

```

16890 \newcommand{\glslongextraLocationSymDescNameTabularFooter}{%
16891   \bottomrule
16892 }

```

c-sym-desc-name Four column style with location list, symbol, description and name.

```

16893 \newglossarystyle{long-loc-sym-desc-name}%
16894 {%
16895   \ifGlsLongExtraUseTabular
16896     \renewenvironment{theglossary}%
16897     {%
16898       \glslongextraSymLocSetDescWidth
16899       \edef\@glslongextra@begintab{%
16900         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16901           \expandonce\glslongextraLocationAlign
16902           \expandonce\glslongextraSymbolAlign
16903           \expandonce\glslongextraDescAlign
16904           \expandonce\glslongextraNameAlign
16905         }}%
16906       \@glslongextra@begintab
16907     }%
16908     {%
16909       \glslongextraLocationSymDescNameTabularFooter
16910       \end{tabular}%
16911     }%
16912     \renewcommand*\glossaryheader{\glslongextraLocationSymDescNameTabularHeader}%
16913   \else
16914     \renewenvironment{theglossary}%
16915     {%
16916       \glspatchLToutput
16917       \glslongextraSymLocSetDescWidth
16918       \edef\@glslongextra@begintab{%
16919         \noexpand\begin{longtable}{%
16920           \expandonce\glslongextraLocationAlign
16921           \expandonce\glslongextraSymbolAlign
16922           \expandonce\glslongextraDescAlign
16923           \expandonce\glslongextraNameAlign
16924         }}%
16925       \@glslongextra@begintab

```

```

16926 }%
16927 {\end{longtable}}}%
16928 \renewcommand*{\glossaryheader}{\glslongextraLocationSymDescNameHeader}%
16929 \fi
16930 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
16931 \renewcommand{\glossentry}[2]{%
16932   \glslongextraLocationFmt{##1}{##2} &
16933   \glslongextraSymbolFmt{##1} &
16934   \glslongextraDescFmt{##1} &
16935   \glslongextraNameFmt{##1}\tabularnewline
16936 }%
16937 \renewcommand{\subglossentry}[3]{%
16938   \glslongextraSubLocationFmt{##1}{##2}{##3} &
16939   \glslongextraSubSymbolFmt{##1}{##2} &
16940   \glslongextraSubDescFmt{##1}{##2} &
16941   \glslongextraSubNameFmt{##1}{##2}\tabularnewline
16942 }%
16943 \ifglsnogroupskip
16944   \renewcommand*{\glsgroupskip}{}%
16945 \else
16946   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
16947 \fi
16948 }

```

scSymNameHeader

```

16949 \newcommand{\glslongextraDescSymNameHeader}{%
16950   \glslongextraDescSymNameTabularHeader\endhead
16951   \glslongextraDescSymNameTabularFooter\endfoot
16952 }

```

meTabularHeader

```

16953 \newcommand{\glslongextraDescSymNameTabularHeader}{%
16954   \toprule
16955   \glslongextraHeaderFmt\descriptionname &
16956   \glslongextraHeaderFmt\symbolname &
16957   \glslongextraHeaderFmt\entryname\tabularnewline
16958   \midrule
16959 }

```

meTabularFooter

```

16960 \newcommand{\glslongextraDescSymNameTabularFooter}{%
16961   \bottomrule
16962 }

```

g-desc-sym-name Three column style with description in the first column, symbol in the second and name in the third.

```

16963 \newglossarystyle{long-desc-sym-name}%
16964 {%
16965   \ifGlsLongExtraUseTabular

```



```

16966 \renewenvironment{theglossary}%
16967 {%
16968     \glslongextraSymSetDescWidth
16969     \edef\@glslongextra@begintab{%
16970         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
16971             \expandonce\glslongextraDescAlign
16972             \expandonce\glslongextraSymbolAlign
16973             \expandonce\glslongextraNameAlign
16974         }}%
16975     \@glslongextra@begintab
16976 }%
16977 {%
16978     \glslongextraDescSymNameTabularFooter
16979     \end{tabular}%
16980 }%
16981 \renewcommand*\glossaryheader{\glslongextraDescSymNameTabularHeader}%
16982 \else
16983 \renewenvironment{theglossary}%
16984 {%
16985     \glspatchLToutput
16986     \glslongextraSymSetDescWidth
16987     \edef\@glslongextra@begintab{%
16988         \noexpand\begin{longtable}{%
16989             \expandonce\glslongextraDescAlign
16990             \expandonce\glslongextraSymbolAlign
16991             \expandonce\glslongextraNameAlign
16992         }}%
16993     \@glslongextra@begintab
16994 }%
16995 {\end{longtable}}%
16996 \renewcommand*\glossaryheader{\glslongextraDescSymNameHeader}%
16997 \fi
16998 \renewcommand*\glsgroupheading[1]{\glslongextraGroupHeading{3}{##1}}%
16999 \renewcommand{\glossentry}[2]{%
17000     \glslongextraDescFmt{##1} &
17001     \glslongextraSymbolFmt{##1} &
17002     \glslongextraNameFmt{##1}\tabularnewline
17003 }%
17004 \renewcommand{\subglossentry}[3]{%
17005     \glslongextraSubDescFmt{##1}{##2} &
17006     \glslongextraSubSymbolFmt{##1}{##2} &
17007     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
17008 }%
17009 \ifglsgroupskip
17010     \renewcommand*\glsgroupskip{}%
17011 \else
17012     \renewcommand*\glsgroupskip{\glspenaltygroupskip}%
17013 \fi
17014 }

```

scSymNameHeader

```
17015 \newcommand{\glslongextraLocationDescSymNameHeader}{%
17016   \glslongextraLocationDescSymNameTabularHeader\endhead
17017   \glslongextraLocationDescSymNameTabularFooter\endfoot
17018 }
```

meTabularHeader

```
17019 \newcommand{\glslongextraLocationDescSymNameTabularHeader}{%
17020   \toprule
17021   \glslongextraHeaderFmt\pagelistname &
17022   \glslongextraHeaderFmt\descriptionname &
17023   \glslongextraHeaderFmt\symbolname &
17024   \glslongextraHeaderFmt\entryname\tabularnewline
17025   \midrule
17026 }
```

meTabularFooter

```
17027 \newcommand{\glslongextraLocationDescSymNameTabularFooter}{%
17028   \bottomrule
17029 }
```

c-desc-sym-name Four column style with location list, description, symbol and name.

```
17030 \newglossarystyle{long-loc-desc-sym-name}%
17031 {%
17032   \ifGlsLongExtraUseTabular
17033     \renewenvironment{theglossary}%
17034     {%
17035       \glslongextraSymLocSetDescWidth
17036       \edef\@glslongextra@begintab{%
17037         \noexpand\begin{tabular}[\glslongextraTabularVAlign]{%
17038           \expandonce\glslongextraLocationAlign
17039           \expandonce\glslongextraDescAlign
17040           \expandonce\glslongextraSymbolAlign
17041           \expandonce\glslongextraNameAlign
17042         }}%
17043       \@glslongextra@begintab
17044     }%
17045     {%
17046       \glslongextraLocationDescSymNameTabularFooter
17047       \end{tabular}%
17048     }%
17049     \renewcommand*\glossaryheader{\glslongextraLocationDescSymNameTabularHeader}%
17050   \else
17051     \renewenvironment{theglossary}%
17052     {%
17053       \glspatchLToutput
17054       \glslongextraSymLocSetDescWidth
17055       \edef\@glslongextra@begintab{%
17056         \noexpand\begin{longtable}{%

```

```

17057         \expandonce\glslongextraLocationAlign
17058         \expandonce\glslongextraDescAlign
17059         \expandonce\glslongextraSymbolAlign
17060         \expandonce\glslongextraNameAlign
17061     } }%
17062     \@glslongextra@begintab
17063 }%
17064 {\end{longtable}}}%
17065 \renewcommand*{\glossaryheader}{\glslongextraLocationDescSymNameHeader}%
17066 \fi
17067 \renewcommand*{\glsgroupheading}[1]{\glslongextraGroupHeading{4}{##1}}%
17068 \renewcommand{\glossentry}[2]{%
17069     \glslongextraLocationFmt{##1}{##2} &
17070     \glslongextraDescFmt{##1} &
17071     \glslongextraSymbolFmt{##1} &
17072     \glslongextraNameFmt{##1}\tabularnewline
17073 }%
17074 \renewcommand{\subglossentry}[3]{%
17075     \glslongextraSubLocationFmt{##1}{##2}{##3} &
17076     \glslongextraSubDescFmt{##1}{##2} &
17077     \glslongextraSubSymbolFmt{##1}{##2} &
17078     \glslongextraSubNameFmt{##1}{##2}\tabularnewline
17079 }%
17080 \ifglsnogroupskip
17081     \renewcommand*{\glsgroupskip}{}%
17082 \else
17083     \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
17084 \fi
17085 }

```

5 topic styles (glossary-topic.sty)

5.1 Package Initialisation and Options

Provides “topic” styles where top-level entries are considered a topic.

```
17086 \NeedsTeXFormat{LaTeX2e}
17087 \ProvidesPackage{glossary-topic}[2020/02/28 v1.43 (NLCT)]
```

Load required package.

```
17088 \RequirePackage{multicol}
```

The top-level entries act like headers. If the top-level entry has a description it's placed below the name.

topic

```
17089 \newglossarystyle{topic}{%
17090   \renewenvironment{theglossary}%
17091   {%
17092     \glstopicInit
17093     \def\glstopic@prechildren{}%
17094     \def\glstopic@prevlevel{-1}%
17095   }%
17096   {\par}%
17097   \renewcommand*{\glossaryheader}{}%
17098   \renewcommand*{\glsgroupheading}[1]{%
17099     \def\glstopic@prevlevel{-1}%
17100     \glstopicGroupHeading{##1}%
17101   }%
17102   \renewcommand{\glossentry}[2]{%
17103     \hangindent0pt\relax
17104     \parindent\glstopicParIndent\relax
17105     \glstopicItem{##1}{##2}%
17106     \ifglshasdesc{##1}%
17107     {%
17108       \def\glstopic@prechildren{}%
17109     }%
17110     {%
17111       \def\glstopic@prechildren{\nopagebreak}%
17112     }%
17113   }%
17114   \renewcommand{\subglossentry}[3]{%
17115     \ifnum\glstopic@prevlevel=0\relax\glstopic@prechildren\fi
```

```

17116 \def\glstopic@prevlevel{##1}%
17117 \glstopicAssignSubIndent{##1}%
17118 \glstopicSubItem{##1}{##2}{##3}%
17119 }%
17120 \renewcommand*{\glsgroupskip}{}%
17121 }

```

picGroupHeading

```
\glstopicGroupHeading{<group label>}
```

May be redefined if letter group headings are required. For example:

```

\renewcommand*{\glstopicGroupHeading}[1]{%
  \glstrgetgrouptitle{#1}{\thisgrptitle}%
  \section*{\thisgrptitle}%
}
17122 \newcommand*{\glstopicGroupHeading}[1]{%

```

\glstopicItem

```
\glstopicItem{<label>}{<location list>}
```

```

17123 \newcommand*{\glstopicItem}[2]{%
17124 \glsparskip\glstopicPreSkip\glsparskip\noindent
17125 \glstopicMarker{#1}%
17126 \glstopicTitleFont
17127 {%
17128 \glstentryitem{#1}\glstarget{#1}{\glstopicTitle{#1}}%
17129 }%
17130 \ifglshasdesc{#1}%
17131 {\glsparskip\nobreak\glstopicMidSkip\glsparskip\nobreak
17132 \@afterheading\glstopicDesc{#1}\glsparskip\glstopicPostSkip}%
17133 {\glsparskip\nobreak\glstopicPostSkip}%
17134 \glstopicLoc{#1}{#2}%
17135 }

```

\glstopicMarker May be used to insert a bookmark etc if required.

```
17136 \newcommand*{\glstopicMarker}[1]{%

```

\glstopicName

```

17137 \newcommand*{\glstopicTitle}[1]{\Glossentryname{#1}%
17138 \ifglshassymbol{#1}{\space\glossentrysymbol{#1}}{}}%
17139 }

```

stopicTitleFont

```
17140 \newcommand*{\glstopicTitleFont}[1]{\textbf{\large #1}}
```

```

\glstopicDesc
17141 \newcommand*{\glstopicDesc}[1]{\Glossentrydesc{#1}\glspostdescription}

\glstopicLoc
17142 \newcommand*{\glstopicLoc}[2]{}

stopicParIndent
17143 \newlength\glstopicParIndent
17144 \setlength\glstopicParIndent{20pt}

stopicSubIndent
17145 \newlength\glstopicSubIndent
17146 \setlength\glstopicSubIndent{20pt}

\glstopicInit
17147 \newcommand{\glstopicInit}{}

```

AssignSubIndent

```
\glstopicAssignSubIndent{<level>}
```

Used to set the indentation for sub-levels.

```

17148 \newcommand*{\glstopicAssignSubIndent}[1]{%

17149   \par
17150   \parindent\dimexpr#1\glstopicSubIndent-\glstopicSubIndent\relax
17151   \glstopicAssignWidest{#1}%
17152   \hangindent\dimexpr\parindent+\glstopicwidest\relax
17153 }

```

\glstopicwidest

```
17154 \newlength\glstopicwidest
```

picAssignWidest

```
\glstopicAssignWidest{<level>}
```

Used in the definition of `\glstopicAssignSubIndent` to set the indentation from the widest name for the given level. This will require glossary-tree to set the values.

```

17155 \newcommand*{\glstopicAssignWidest}[1]{%
17156   \ifcsundef{@glswidestlength\romannumeral#1}%
17157   {%
17158     \ifcsdef{@glswidestname\romannumeral#1}%
17159     {%
17160       \settowidth{\glstopicwidest}{%

```

```

17161      \glstopicSubNameFont{\csuse{@glswidestname\romannumeral#1}}}%
17162      \glstopicSubItemSep
17163      }%
17164      }%
17165      {\setlength{\glstopicwidest}{0pt}}}%

    Save the value so that it doesn't have to keep being recalculated.

17166      \csedef{@glswidestlength\romannumeral#1}{\the\glstopicwidest}%
17167      }%
17168      {\setlength{\glstopicwidest}{\csuse{@glswidestlength\romannumeral#1}}}%
17169      }

```

glstopicPreSkip

```
17170 \newcommand*{\glstopicPreSkip}{\medskip}
```

glstopicMidSkip

```
17171 \newcommand*{\glstopicMidSkip}{\smallskip}
```

lstopicPostSkip

```
17172 \newcommand*{\glstopicPostSkip}{\smallskip}
```

glstopicSubItem

```
\glstopicSubItem{<level>}{<label>}{<location list>}
```

```

17173 \newcommand*{\glstopicSubItem}[3]{%
17174   \glstopicSubItemBox{#1}{\glstopicSubNameFont{\glstentryitem{#2}}%
17175     \glstarget{#2}{\glossentryname{#2}}}%
17176   \glstopicSubItemSep
17177   }%
17178   \ifglshassymbol{#2}{(\glossentrysymbol{#2})\space}{}%

17179   \ifglshasdesc{#2}%
17180     {\glossentrydesc{#2}\glspostdescription\glstopicSubPreLocSep}{}%
17181   \glstopicSubLoc{#2}{#3}%
17182 }

```

topicSubItemSep

```
17183 \newcommand*{\glstopicSubItemSep}{\quad}
```

topicSubItemBox

```
\glstopicSubItemBox{<level>}{<text>}
```

```

17184 \newcommand*{\glstopicSubItemBox}[2]{%
17185   \ifdim\glstopicwidest>0pt\relax\makebox[\glstopicwidest][l]{#2}\else#2\fi
17186 }

```

opicSubNameFont

```
17187 \newcommand*{\glstopicSubNameFont}[1]{\textbf{#1}}
```

picSubPreLocSep

```
17188 \newcommand*{\glstopicSubPreLocSep}{\space}
```

\glstopicSubLoc

```
17189 \newcommand*{\glstopicSubLoc}[2]{#2}
```

\glstopicCols

```
17190 \newcommand*{\glstopicCols}{2}
```

glstopicColsEnv

```
17191 \newcommand*{\glstopicColsEnv}{multicols}
```

topicmcols

```
17192 \newglossarystyle{topicmcols}{%
17193   \renewenvironment{theglossary}%
17194   {%
17195     \glstopicInit
17196     \def\glstopic@prechildren{}%
17197     \def\glstopic@postchildren{}%
17198     \def\glstopic@prevlevel{-1}%
17199   }%
17200   {%
17201     \ifnum\glstopic@prevlevel>0\relax\glstopic@postchildren\fi
17202     \par
17203   }%
17204   \renewcommand*{\glossaryheader}{}%
17205   \renewcommand*{\glsgroupeheading}[1]{%
17206     \ifnum\glstopic@prevlevel>0\relax\glstopic@postchildren\fi
17207     \def\glstopic@prevlevel{-1}%
17208     \glstopicGroupHeading{##1}%
17209   }%
17210   \renewcommand{\glossentry}[2]{%
17211     \ifnum\glstopic@prevlevel>0\relax\glstopic@postchildren\fi
17212     \def\glstopic@prevlevel{0}%
17213     \hangindent0pt\relax
17214     \parindent\glstopicParIndent\relax
17215     \glstopicItem{##1}{##2}%
17216     \ifnum\glstopicCols>1\relax
```

If there isn't a description, penalise a page break.

```
17217     \ifglshasdesc{##1}%
17218     {%
17219       \edef\glstopic@prechildren{%
17220         \noexpand\begin{\glstopicColsEnv}{\glstopicCols}}%
17221       }%
17222     }%
```



```

17223     \edef\glstopic@prechildren{%
17224         \noexpand\nopagebreak
17225         \noexpand\begin{\glstopicColsEnv}{\glstopicCols}}%
17226     }%
17227     \edef\glstopic@postchildren{\noexpand\end{\glstopicColsEnv}}%
17228     \fi
17229 }%
17230 \renewcommand{\subglossentry}[3]{%
17231     \ifnum\glstopic@prevlevel=0\relax\glstopic@prechildren\fi
17232     \def\glstopic@prevlevel{##1}%
17233     \glstopicAssignSubIndent{##1}%
17234     \glstopicSubItem{##1}{##2}{##3}%
17235 }%
17236 \renewcommand*{\glsgroupskip}{}%
17237 }

```

Glossary

bib2gls A command line Java application that selects entries from a .bib file and converts them to glossary definitions (like bibtex but also performs hierarchical sorting and collation, thus omitting the need for **xindy** or **makeindex**). Further details at: <http://www.dickimaw-books.com/software/bib2gls/..>

First use The first time a glossary entry is used (from the start of the document or after a reset) with one of the following commands: `\gls`, `\Gls`, `\GLS`, `\glspl`, `\Glspl`, `\GLSpl` or `\glsdisp`. *see* **First use flag** & **First use text**

First use flag A conditional that determines whether or not the entry has been used according to the rules of **first use**.

First use text The text that is displayed on **first use**, which is governed by the first and first-plural keys of `\newglossaryentry`. (May be overridden by `\glsdisp`.)

makeindex An indexing application.

xindy An flexible indexing application with multilingual support written in Perl.

Change History

0.1 (2015-11-22)	
General: Initial experimental release	5
0.2 (2015-11-30)	
\Glsfmtshort: new	361
\glsfmtshort: new	360
\Glsfmtshortpl: new	361
\glsfmtshortpl: new	360
short: switched inline full form to short (long)	252
0.3 (2015-12-02)	
\@ACRlong: added redefinition	91
\@ACRlongpl: added redefinition	92
\@ACRshort: added redefinition	89
\@ACRshortpl: added redefinition	90
\@Acrlong: added redefinition	91
\@Acrlongpl: added redefinition	92
\@Acrshort: added redefinition	89
\@Acrshortpl: added redefinition	90
\@GLSdesc: added redefinition	85
\@GLSdescplural@: added redefinition	85
\@GLSfirst@: added redefinition	82
\@GLSfirstplural@: added redefinition	84
\@GLSname@: added redefinition	84
\@GLSplural@: added redefinition	83
\@GLSsymbol@: added redefinition	86
\@GLSsymbolplural@: added redefinition	86
\@GLStext@: added redefinition	81
\@GLSuseri@: added redefinition	86
\@GLSuserii@: added redefinition	87
\@GLSuseriii@: added redefinition	87
\@GLSuseriv@: added redefinition	87
\@GLSuserv@: added redefinition	88
\@GLSuservi@: added redefinition	88
\@Glsdesc@: added redefinition	84
\@Glsdescplural@: added redefinition	85
\@Glsfirst@: added redefinition	82
\@Glsfirstplural@: added redefinition	83
\@Glsname@: added redefinition	84
\@Glsplural@: added redefinition	83
\@Glsymbol@: added redefinition	85
\@Glsymbolplural@: added redefinition	86
\@Glsxtr@defaultnoglossarywarning: new	149
\@Glsxtr@field@linkdefs: new	80
\@Glsxtr@insertdots: new	216
\@print@glossary: added redefinition	145
\glsabbrvdefaultfont: renamed from \abbrvdefaultfont	222
\glsaccessdesc: new	176
\glsaccessdescplural: new	176
\glsaccessfirst: new	173
\glsaccessfirstplural: new	174
\glsaccesslong: new	178
\glsaccesslong: new	178
\glsaccessname: new	172
\glsaccessplural: new	173
\glsaccessshort: new	177
\glsaccessshort: new	177
\glsaccessshortpl: new	178

\glsaccesssshortpl: new	177	\cGLSpl: new	120
\glsaccessssymbol: new	174	\cGLSpl@: new	120
\glsaccessssymbolplural: new	175	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	172	new	115
\glstentryfmt: added check for short ..	72	\cGLS: new	119
\glslongpltok: new	216	\cGLSformat: new	119
\glsshortpltok: new	216	\cGLSpl: new	119
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	120
name in \setkeys	218	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	18
for plural	213	\glsenableentrycount: new	115
\GLSxtrlongpl: new	233	\glsfirstabbrvdefaultfont: new ..	222
\Glsxtrlongpl: new	232	\glsfirstlongdefaultfont: new ...	223
\glsxtrlongpl: new	232	\Glsfmtfirst: new	364
\glsxtrNoGlossaryWarning: new	24	\glsfmtfirst: new	364
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	365
new	213	\glsfmtfirstpl: new	365
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	363
new	213	\glsfmtplural: new	363
\glsxtrpostlinkendsentence: new ..	212	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	231	\Glsxtrtitleshort	361
\Glsxtrshortpl: new	231	renamed from \Glstentryfmtshort ..	361
\glsxtrshortpl: new	230	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort	360
\glslabeltok	244	renamed from \glstentryfmtshort ..	360
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	242	\Glsxtrtitleshortpl	361
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl	361
redefinition of \acronymtype	20	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl	360
\Glsxtrshort	361	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl	360
\glsxtrshort	360	\Glsfmttext: new	362
\Glsfmtshortpl: changed to use		\glsfmttext: new	362
\glsxtrshortpl	361	\glshasattribute: new	190
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	189
\glsxtrshortpl	360	\glsxtremsuffix: new	293
\glsxtrifemptyglossary: new	32	\GlsXtrEnableEntryCounting: new ..	114
\glsxtrnewnumber: added extra		\glsxtrifcounttrigger: new	117
argument	194	\glsxtrscfont: new	259
\glsxtrnewsymbol: added extra		\glsxtrscsuffix: new	260
argument	193	\glsxtrsmfont: new	276
\MakeAcronymsAbbreviations: set the		\glsxtrsmsuffix: new	276
default type to \acronymtype	129	short-em: new	301
\newterm: fixed name argument	193	short-em-desc: new	302
0.5 (2015-12-07)		short-em-footnote: new	312
\@cGLS: new	119	short-em-long: new	297
\@cGLS@: new	119	short-em-long-desc: new	298

short-em-postfootnote: new	315	\glxtrheadshortpl: now uses headuc	
short-sc-footnote: new	271	attribute	350
short-sc-postfootnote: new	273	\Glsxtrheadtext: now uses headuc	
short-sm: new	280	attribute	353
short-sm-desc: new	282	\glxtrheadtext: now uses headuc	
short-sm-footnote: new	287	attribute	352
short-sm-long: new	278	short-em-footnote: switch off regular	
short-sm-long-desc: new	280	attribute if set	313
short-sm-postfootnote: new	290	short-em-footnote-desc: switch off	
long-noshort-em: new	305	regular attribute if set	314
long-noshort-em-desc: new	309	short-long: switch off regular attribute	
long-noshort-sm: new	284	if set	243
long-noshort-sm-desc: new	286	short-long-desc: switch off regular	
long-short-em: new	293	attribute if set	244
long-short-em-desc: new	294	short-postfootnote-desc: switch off	
long-short-sm: new	277	regular attribute if set	250
long-short-sm-desc: new	278	short-sc-footnote: switch off regular	
0.5.1 (2015-12-02)		attribute if set	271
\Glsaccessstext: new	172	short-sc-footnote-desc: switch off	
0.5.1 (2015-12-07)		regular attribute if set	273
\@glxtr@doaccsupp: new	23	short-sm-footnote: switch off regular	
General: removed \ifglxtruseuchead	349	attribute if set	288
\Glsaccessdesc: new	176	short-sm-footnote-desc: switch off	
\Glsaccessdescplural: new	176	regular attribute if set	290
\Glsaccessfirst: new	173	long-short: switch off regular attribute	
\Glsaccessfirstplural: new	174	if set	241
\Glsaccessname: new	172	long-short-desc: switch off regular	
\Glsaccessplural: new	173	attribute if set	242
\Glsaccesssymbol: new	175	long-short-sc-desc: switch off regular	
\Glsaccesssymbolplural: new	175	attribute if set	261
\Glsxtrheadfirst: now uses headuc		footnote: switch off regular attribute if	
attribute	355	set	246
\glxtrheadfirst: now uses headuc		postfootnote: switch off regular	
attribute	354	attribute if set	248
\Glsxtrheadfirstplural: now uses		0.5.2 (2015-12-08)	
headuc attribute	355	\@GLSdesc@: added accessibility support	85
\glxtrheadfirstplural: now uses		\@GLSdescplural@: added accessibility	
headuc attribute	355	support	85
\Glsxtrheadplural: now uses headuc		\@GLSfirst@: added accessibility	
attribute	354	support	82
\glxtrheadplural: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	353	support	84
\Glsxtrheadshort: now uses headuc		\@GLSname@: added accessibility support	84
attribute	350	\@GLSplural@: added accessibility	
\glxtrheadshort: now uses headuc		support	83
attribute	349	\@GLSsymbol@: added accessibility	
\Glsxtrheadshortpl: now uses headuc		support	86
attribute	351	\@GLSsymbolplural@: added	
		accessibility support	86

\@GLStext@: added accessibility support	81	\GLSaccesslongpl: new	179, 188
\@Glsdesc@: added accessibility support	84	\Glsaccesslongpl: new	178
\@Glsdescplural@: added accessibility support	85	\glsaccesslongpl: new	178
\@Glsfirst@: added accessibility support	82	\GLSaccessname: new	172, 185
\@Glsfirstplural@: added accessibility support	83	\GLSaccessplural: new	173, 186
\@Glsname@: add accessibility support	84	\GLSaccessshort: new	177, 187
\@Glsplural@: added accessibility support	83	\GLSaccessshortpl: new	178, 188
\@Glsymbol@: added accessibility support	85	\GLSaccesssymbol: new	175, 186
\@Glsymbolplural@: added accessibility support	86	\GLSaccesssymbolplural: new	175, 187
\@Glstext@: added accessibility support	81	\GLSaccessstext: new	172, 185
\@Glsdesc@: added accessibility support	84	\glsentryfmt: moved	
\@Glsdescplural@: added accessibility support	85	\glssetabbrvfmt from	
\@Glsfirst@: added accessibility support	82	\glsxtrabbrvfmt to here	72
\@Glsfirstplural@: added accessibility support	83	\GlsXtrEnableInitialTagging: new	209
\@Glsname@: added accessibility support	84	\glsxtrfieldtitlecase: new	194
\@Glsplural@: added accessibility support	82	\GlsXtrFormatLocationList: new	70
\@Glsymbol@: added accessibility support	85	\glsxtrnewabbrevpresetkeyhook:	
\@Glsymbolplural@: added accessibility support	86	new	221
\@Glstext@: added accessibility support	81	\glsxtrtagfont: new	210
\@Glsdesc@: added accessibility support	84	\KV@printgloss@nonumberlist: added	71
\@Glsdescplural@: added accessibility support	85	\mfu@checkword@do: added	209
\@Glsfirst@: added accessibility support	82	\setabbreviationstyle: added check	
\@Glsfirstplural@: added accessibility support	83	for post-definition style switch	237
\@Glsname@: added accessibility support	84	0.5.3 (2015-12-09)	
\@Glsplural@: added accessibility support	82	\@glsxtr@autoindex@at: new	205
\@Glsymbol@: added accessibility support	85	\@glsxtr@autoindex@encap: new	206
\@Glsymbolplural@: added accessibility support	86	\@glsxtr@autoindex@esc: new	206
\@Glstext@: added accessibility support	81	\@glsxtr@autoindex@level: new	206
\@glsxtr@activate@initialtagging:		\@glsxtr@autoindex@setname: new	204
new	210	\@glsxtr@doabbreviationsdef: new	20
\@glsxtr@do@titlecaps@warn: new	210	General: removed	
\@glsxtr@tag: new	210	\GlsXtrNoGlsWarningNoAutoMakeMain	147
General: fixed typo in glossaries-accsupp		\glsdescwidth: added	69
and tidied up code to use just one		\glspagelistwidth: added	69
\@ifpackageloaded	171	\glsxtrdoautoindexname: new	203
removed \glsxtrabbrvfmt	234	\glsxtrpostnamehook: new	200
\glossaryentrynumbers: added	69	\if@glsxtr@format@override: new	202
\Glossentrydesc: added	207	\ProvidesGlossariesExtraLang: new	369
\Glossentryname: added	199	\RequireGlossariesExtraLang: new	369
\Glossentrysymbol: added	208	0.5.4 (2015-12-15)	
\glossentrysymbol: added	207	\@newglossaryentry@defunitcounters:	
\GLSaccessdesc: new	176, 187	new	120
\GLSaccessdescplural: new	177, 187	\@GLSxtr@p@acrlong@: new	106
\GLSaccessfirst: new	174, 186	\@GLSxtr@p@acrlongpl@: new	106
\GLSaccessfirstplural: new	174, 186	\@GLSxtr@p@acrshort@: new	105
\GLSaccesslong: new	178, 188	\@GLSxtr@p@acrshortpl@: new	106
		\@GLSxtr@p@long@: new	105
		\@GLSxtr@p@longpl@: new	105

\@GLSxtr@p@plural@: new	104	\@sGlsXtrEnableOnTheFly: new	65
\@GLSxtr@p@short@: new	104	\cGlsformat: added	120
\@GLSxtr@p@shortpl@: new	105	\cGlsformat: added	120
\@GLSxtr@p@text@: new	104	\cGlsplformat: added	120
\@GlsXtrEnableOnTheFly: new	65	\cGlsplformat: added	120
\@Glsxtr: new	66	\gl:disablehyper: added	102
\@Glsxtr@p@acrlong@: new	106	\gl:donohyperlink: added	102
\@Glsxtr@p@acrlongpl@: new	106	\gl:enableentryunitcount: new ...	123
\@Glsxtr@p@acrshort@: new	105	\gl:shasattribute: added check for	
\@Glsxtr@p@acrshortpl@: new	105	entry's existence	190
\@Glsxtr@p@long@: new	105	\gl:sifattribute: added check for	
\@Glsxtr@p@longpl@: new	105	entry's existence	191
\@Glsxtr@p@plural@: new	104	\gl:postlinkhook: added existence	
\@Glsxtr@p@short@: new	104	check	212
\@Glsxtr@p@shortpl@: new	104	\Glsxtr: new	66
\@Glsxtr@p@text@: new	103	\gl:xtr: new	65
\@Glsxtrpl: new	67	\gl:xtrcat: new	65
\@alt@gl:shyp@opt: new	99	\gl:xtrdohyperlink: added	101
\@gl:alt@hyp@opt: new	99	\gl:xtrdowrglossaryhook: new	99
\@gl:alt@hyp@opt@char: new	99	\GlsXtrEnableEntryUnitCounting:	
\@gl:alt@hyp@opt@keys: new	99	new	126
\@gl:increment@currunitcount:		\GlsXtrEnableOnTheFly: new	64
new	121	\Glsxtrpl: new	66
\@gl:local@increment@currunitcount:		\gl:xtrpl: new	66
new	122	\gl:xtrpostlocalreset: new	114
\@gl:setdefault@gl:link@opts:		\gl:xtrpostlocalunset: new	113
new	97	\gl:xtrpostreset: new	113
\@gl:xtr: new	65	\gl:xtrpostunset: new	112
\@gl:xtr@addunitcounter: new	121	\gl:xtrprotectlinks: new	103
\@gl:xtr@currunitcount: new	122	\GlsXtrSetAltModifier: new	99
\@gl:xtr@ifunitcounter: new	121	\GlsXtrSetDefaultGlsOpts: new	98
\@gl:xtr@p@acrlong@: new	106	\gl:xtrstarflywarn: new	65
\@gl:xtr@p@acrlongpl@: new	106	\GlsXtrWarning: new	67
\@gl:xtr@p@acrshort@: new	105	\MakeAcronymsAbbreviations: now	
\@gl:xtr@p@acrshortpl@: new	105	disables \setacronymstyle	129
\@gl:xtr@p@long@: new	105	1.0 (2016-01-24)	
\@gl:xtr@p@longpl@: new	105	\@gl:xtr@autoindexcrossrefs: new .	18
\@gl:xtr@p@plural@: new	104	\@gl:xtr@idx@displaynumberlist:	
\@gl:xtr@p@short@: new	104	new	139
\@gl:xtr@p@shortpl@: new	104	\@gl:xtr@idx@entrynumberlist: new	140
\@gl:xtr@p@text@: new	103	\@gl:xtr@noidx@displaynumberlist:	
\@gl:xtr@prevunitcount: new	122	new	139
\@gl:xtr@setentryunitcountunsetattr:		\@gl:xtr@noidx@entrynumberlist:	
new	126	new	140
\@gl:xtr@unitcountlist: new	121	\@gl:xtr@noidx@numberlistloop:	
\@gl:xtrpl: new	66	new	139
\@newglossaryentryposthook: added		\@gl:xtr@reg@glosslist: new	131
empty see value if not set and added		\makeglossaries: new	131
'see' to field key map	53		

1.01 (2016-02-02)	
\glxtrdiscardperiod: added check for first use	213
short-desc: fixed typo in \glxtrininlinefullformat and added missing second argument ...	253
1.02 (2016-04-25)	
\@glxtr@current@style: new	68
\Glsfmtfull: new	368
\Glsfmtfull: new	367
\Glsfmtfullpl: new	368
\Glsfmtfullpl: new	368
\Glsfmtlong: new	366
\Glsfmtlong: new	365
\Glsfmtlongpl: new	367
\Glsfmtlongpl: new	366
\Glsxtrheadfull: new	359
\Glsxtrheadfullpl: new	358
\Glsxtrheadfullpl: new	359
\Glsxtrheadfullpl: new	358
\Glsxtrheadlong: new	357
\Glsxtrheadlong: new	356
\Glsxtrheadlongpl: new	357
\Glsxtrheadlongpl: new	356
\Glsxtrtitlefull: new	359
\Glsxtrtitlefull: new	358
\Glsxtrtitlefullpl: new	360
\Glsxtrtitlefullpl: new	359
\Glsxtrtitlelong: new	357
\Glsxtrtitlelong: new	356
\Glsxtrtitlelongpl: new	358
\Glsxtrtitlelongpl: new	357
short-postfootnote-desc: added redef of \glxtrsetupfulldefs ..	251
\ifglxtrininsertinside: new	240
postfootnote: added redef of \glxtrsetupfulldefs	249
stylemods: new	24
1.03 (2016-04-27)	
\@GLSfirstplural@: bug fix: misspelt cs name	84
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	83
\@GLSfirstplural@: bug fix: misspelt cs name	83
\@GLsplural@: fixed bug \@GLsplural@ should be redefined not \@GLsplural	83
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	82
\glxtrtitlelongpl: bug fix: changed \glxtrlong to \glxtrlongpl ..	357
\glxtrtitleshortpl: bug fix: changed \glxtrshort to \glxtrshortpl	350
1.04 (2015-04-30)	
short-em-footnote: renamed from “footnote-em”	312
1.04 (2016-05-02)	
\@@glxtrpostloctag: new	71
\@GLSdesc@: set abbreviation and regular format	85
\@GLSdescplural@: set abbreviation and regular format	85
\@GLSfirst@: set abbreviation format ..	82
\@GLSfirstplural@: set abbreviation and regular format	84
\@GLSname@: set abbreviation and regular format	84
\@GLSplural@: set abbreviation and regular format	83
\@GLSsymbol@: set regular format	86
\@GLSsymbolplural@: set regular format	86
\@GLStext@: set abbreviation and regular format	81
\@GLSuseri@: set regular format	86
\@GLSuserii@: set regular format	87
\@GLSuseriii@: set regular format	87
\@GLSuseriv@: set regular format	87
\@GLSuseriv@: set regular format	88
\@GLSuservi@: set regular format	88
\@GLSdesc@: set abbreviation and regular format	84
\@GLSdescplural@: set abbreviation and regular format	85
\@GLSfirst@: set abbreviation and regular format	82
\@GLSfirstplural@: set abbreviation and regular format	83
\@GLSname@: set abbreviation and regular format	84
\@GLsplural@: set abbreviation and regular format	83
\@GLssymbol@: set regular format	85
\@GLssymbolplural@: set regular format	86
\@GLStext@: set abbreviation and regular format	81
\@GLSuseri@: set regular format	86
\@GLSuserii@: set regular format	87
\@GLSuseriii@: set regular format	87

\@Glsuseriv@: set regular format	87	\glslongfont: new	222
\@Glsuserv@: set regular format	88	\glslonguserfont: new	318
\@Glsuservi@: set regular format	88	\glstrassignfieldfont: new	81
\@gls@preglossaryhook: added check for entry's existence	210	\GlsXtrEnablePreLocationTag: new	70
\@glsdesc@: set abbreviation and regular format	84	\glstrfirstscfont: new	260
\@glsdescplural@: set abbreviation and regular format	85	\glstrfirstsmfont: new	276
\@glsfirst@: set abbreviation and regular format	82	\glstrlongshortdescsort: new	241
\@glsfirstplural@: set abbreviation and regular format	83	\glstrpostnamehook: added category check	200
\@glsname@: set abbreviation and regular format	84	\glstrregularfont: new	72
\@glsplural@: set abbreviation and regular format	82	\glstruserfield: new	318
\@glsymbol@: set regular format	85	\glstruserparen: new	318
\@glsymbolplural@: set regular format	86	\glstrusersuffix: new	318
\@glstext@: set abbreviation and regular format	81	\GlsXtrWarnDeprecatedAbbrStyle: new	239
\@glstr@deprecated@abbrstyle: new	239	short-em-long-em: new	299
\@glstr@do@style: new	25	short-em-long-em-desc: new	300
\@glstr@dolocktag: new	71	short-em-nolong: new	302
\@glstr@idx@entrynumberlist: switched from \let to \newcommand	140	short-em-nolong-desc: new	304
\@glstr@pagetag: new	71	short-em-postfootnote: renamed from "postfootnote-em"	315
\@glstr@pagetag: new	71	short-footnote: new	247
\@glstr@prelocktag: new	71	short-long-user: new	326
\@glstr@postlocktag: new	71	short-long-user-desc: new	327
\@glstr@prelocktag: new	70, 71	short-nolong: new	253
\glossentrydesc: added glossdescfont attribute check	195	short-nolong-desc: new	255
\Glossentryname: added glossnamefont attribute check	199	short-postfootnote: new	250
\glossentryname: added glossnamefont attribute check	197	short-sc-footnote: renamed from "footnote-sc"	271
moved post name hook inside condition	199	short-sc-nolong: new	265
\glsabbrvmfont: new	293	short-sc-nolong-desc: new	267
\glsabbrvuserfont: new	318	short-sc-postfootnote: renamed from "postfootnote-sc"	273
\glsfirstabbrvmfont: new	293	short-sm-footnote: renamed from "footnote-sm"	287
\glsfirstabbrvuserfont: new	318	short-sm-nolong: new	282
\glsfirstlongemfont: new	293	short-sm-nolong-desc: new	283
\glsfirstlonguserfont: new	318	short-sm-postfootnote: renamed from "postfootnote-sm"	290
\glsifnotregularcategory: new	191	\letabbreviationstyle: new	239
\glslongdefaultfont: new	223	\newabbreviationstyle: bug fix: corrected test for existence	238
\glslongemfont: new	293	long-em-noshort-em: new	306
		long-em-noshort-em-desc: new	310
		long-em-short-em: new	295
		long-em-short-em-desc: new	296
		long-noshort: new	259
		long-noshort-desc: new	258

long-noshort-em: renamed from “long-em”	305	\glxtrComputeTreeIndent: new ...	445
long-noshort-em-desc: renamed from “long-desc-em”	309	\glxtrComputeTreeSubIndent: new	445
long-noshort-sc: renamed from “long-sc”	267	\glxtrtreetopindent: new	436
long-noshort-sc-desc: renamed from “long-desc-sc”	269	short-em-long: fixed incorrect font used by long form	298
long-noshort-sm: renamed from “long-sm”	284	\xglsssetwidest: new	436
long-noshort-sm-desc: renamed from \long-desc-sm	286	1.06 (2016-06-18)	
long-short-user: new	319	\@glstdoifexistsorwarn: new	17
long-short-user-desc: new	325	\@glxtr@docdefval: new	17
\renewabbreviationstyle: new	238	\@glxtr@usesee: new	54
style: new	25	General: disabled docdef key at the start of the document	31
1.05 (2016-06-10)		docdef option changed to choice	16
\eglssetwidest: new	436	\glxtr@usesee: new	54
\glsFindWidestAnyName: new	439	\glxtrusesee: new	54
\glsFindWidestAnyNameLocation: new	444	\glxtruseseeformat: new	54
\glsFindWidestAnyNameSymbol: new	442	\if@glxtrdocdefrestricted: new ..	17
\glsFindWidestAnyNameSymbolLocation: new	443	1.07 (2016-08-15)	
\glsFindWidestLevelTwo: new	440	\@@glxtrp: new	106
\glsFindWidestUsedAnyName: new ..	438	\@GLSfirst@: added check for nohyperfirst attribute	82
\glsFindWidestUsedAnyNameLocation: new	443	\@GLSfirstplural@: added check for nohyperfirst attribute	84
\glsFindWidestUsedAnyNameSymbol: new	441	\@GLSxtrp: new	107
\glsFindWidestUsedAnyNameSymbolLocation: new	442	\@GLSfirst@: added check for nohyperfirst attribute	82
\glsFindWidestUsedLevelTwo: new .	439	\@GLSfirstplural@: added check for nohyperfirst attribute	83
\glsFindWidestUsedTopLevelName: new	438	\@GLSxtrp: new	107
\glsfirstlongfootnotefont: new ..	245	\@gls@preglossaryhook: added \glossxtrsetpopts	211
\glsgetwidestname: new	437	\@GLSfirst@: added check for nohyperfirst attribute	82
\glsgetwidestsubname: new	437	\@GLSfirstplural@: added check for nohyperfirst attribute	83
\glslongfootnotefont: new	245	\@glxtrinmark: new	347
\glxtrAltTreeIndent: new	435	\@glxtrnotinmark: new	347
\glxtralttreeInit: new	436	\@glxtrp: new	107
\glxtrAltTreePar: new	435	\@glxtrp@opt: new	106
\glxtrAltTreeSetHangIndent: new	445	\glossxtrsetpopts: new	106
\glxtrAltTreeSetSubHangIndent: new	445	\glsp: new	109
\glxtralttreeSubSymbolDescLocation: new	436	\glsp: new	109
\glxtralttreeSymbolDescLocation: new	435	\glxtr@entry@p: new	108
		\glxtrabbrvfootnote: new	245
		\glxtrchecknohyperfirst: new	82
		\glxtrfieldtitlecasecs: new	195
		\glxtrifinmark: new	347
		\GLSxtrp: new	110
		\Glsxtrp: new	109

\glxtrp: new	108	\glxtrassignfieldfont: added check for existence	81
\glxtrsetpopts: new	106	\glxtrresourcefile: new	150
short-long-desc: added text key	244	\printunsrtglossaries: new	157
fixed misspelling of \glsabbrvfont in plural key	244	\printunsrtglossary: new	156
long-short-desc: added missing text key	242	1.09 (2016-12-16)	
fixed misspelling of \glsabbrvfont .	242	\@glxtr@gettype: new	138
footnote: changed first forms to use \glsfirstlongfootnotefont ...	245	\@glxtr@mixed@assign@sortkey: new	138
postfootnote: removed \footnote from first keys	248	\@printglossary: redefined to save options	137
switched from \glsfirstlongfont to \glsfirstlongfootnotefont ...	249	\glxtr@makeglossaries: new	138
\RestoreAcronyms: modified \@gls@link@checkfirsthyper to set \glxtrifwasfirstuse	130	1.10 (2016-12-17)	
1.08 (2016-12-13)		\@GLSpl@: fixed bug caused by typo in command name	74
\@@glxtr@record: new	8	1.11 (2017-01-19)	
\@GLS@: added \@glxtr@record	74	\@glxtr@do@redef@forglsentries: new	6
\@GLSpl@: added \@glxtr@record ...	74	\@glxtr@noidx@do: new	161
\@Gls@: added \@glxtr@record	73	\@glxtr@redef@forglsentries: new .	6
\@GLspl@: added \@glxtr@record ...	73	\@glxtr@shortcutsval: new	22
\@gls@: added \@glxtr@record	73	\@glxtr@unsrt@getgrouptitle: new	159
\@gls@link@: added \@glxtr@record	74	\@print@noidx@glossary: added redefinition	142
\@gls@field@link: added \@glxtr@record	73	\glxtr@addloclistfield: added group key	14
\@gls@saveentrycounter: new	30	added location key	14
\@glsdisp: added \@glxtr@record ..	74	\glxtr@fields: new	151
\@glspl@: added \@glxtr@record ...	73	\glxtr@linkprefix: new	152
\@glxtr@dorecord: new	10	\glxtr@org@newignoredglossary: new	49
\@glxtr@err@undefaction: new	6	\glxtr@s@newignoredglossary: new	49
\@glxtr@record: new	7	\glxtr@shortcutsval: new	152
\@glxtr@warn@onexistsordo: new ...	6	\glxtr@texencoding: new	151
\@glxtr@warn@undefaction: new	6	\glxtr@writefields: new	152
\@print@unsrt@glossary: new	157	\GlsXtrLoadResources: new	151
record: added record package option ...	15	\glxtrpageref: new	46
\glsadd: added \@glxtr@record	80	\glxtrresourcefile: changed extension to .glstex	150
\glsdoifexists: now defines \glslabel	52	\newignoredglossary: added starred version	49
\glxtr@@do@wrglossary: new	30	1.12 (2017-02-03)	
\glxtr@addloclistfield: new	13	\@@glxtr@recordcounter: new	13
\glxtr@indexonly@saveentrycounter: new	13	\@gls@preglossaryhook: check for definition	210
\glxtr@record: new	154	\@glxtr@counterrecordhook: new .	154
\glxtr@resource: new	151	\@glxtr@display@loc: new	143
\glxtr@saveentrycounter: new	30	\@glxtr@docounterrecord: new ...	154
\glxtr@setup@record: new	13		

\@glxstr@longnewglossaryentry:	\glxstrfieldifinlist: new	36
new	\glxstrfieldlistadd: new	35
\@glxstr@noop@recordcounter: new	\glxstrfieldlistadd: new	36
13	\glxstrfieldlistgadd: new	36
\@glxstr@op@recordcounter: new	\glxstrfieldlistxadd: new	36
13	\glxstrfieldxifinlist: new	37
\@glxstr@provide@storagekey: new	\glxstrfmt: new	34
33	\GlsXtrFmtDefaultOptions: new	34
\@glxstr@s@longnewglossaryentry:	\GlsXtrFmtField: new	33
new	\glxstrifkeydefined: new	32
48	\glxstrindexaliased: new	98
\@glxstr@entryfmt: new	\GlsXtrLetField: new	41
35	\GlsXtrLetFieldToField: new	41
\@glxstr@indexaliased: new	\GlsXtrLoadResources: removed	
97	restriction on only one per document	151
\@glxstr@setaliasnoindex: new	\glxstrlocrangefmt: new	144
97	\glxstrpostlongdescription: new	49
\@newglossaryentryposthook: added	\glxstrprovidestoragekey: new	32
check for alias key	\GlsXtrRecordCounter: new	154
60	\glxstrresourcecount: new	151
\@no@glxstr@indexaliased: new	\glxstrresourcefile: added catcode	
98	change for @	150
\@printunsrtglossary: new	\glxstrsetaliasnoindex: new	97
156	\GlsXtrSetField: new	41
General: added target key to printgloss	\glxstrsetfieldifexists: new	41
family	\glxstrunsrtdo: new	160
137	\GlsXtrusefield: new	40
\apptoglossarypreamble: new	\glxstrusefield: new	40
46	short-postlong-user: new	322
\csGlsXtrLetField: new	short-postlong-user-desc: new	324
41	\longnewglossaryentry: added starred	
\eGlsXtrSetField: new	version	48
42	long-postshort-user: new	320
\gGlsXtrSetField: new	long-postshort-user-desc: new	322
41	postdot: new	19
\glxnoidxdisplayloc: added	\pretoglossarypreamble: new	46
redefinition	\print@noop@unsrtglossaryunit:	
143	new	159
\glssettocitle: added patch	\print@op@unsrtglossaryunit: new	159
50	\printunsrtglossary: added starred	
\glxstr@counterrecord: new	form	156
154	\printunsrtglossaryhandler: new	158
\glxstr@langtag: new	\printunsrtglossaryunit: new	13
152	\printunsrtglossaryunitsetup: new	159
\glxstr@newabbreviation: new	\provideignoredglossary: new	50
218	\s@glxstr@provide@storagekey: new	33
\glxstr@org@newignoredglossary:	\s@printunsrtglossary: new	157
Added check for existence	\xGlsXtrSetField: new	42
49		
\glxstr@pluralsuffixes: new		
152		
\glxstr@provideignoredglossary:		
new		
50		
\glxstr@s@newignoredglossary:		
Added check for existence		
49		
\glxstr@s@provideignoredglossary:		
new		
51		
\glxstr@abbrvpluralsuffix: new		
223		
\glxstr@alias: new		
60		
\glxstr@copytoglossary: new		
51		
\glxstr@deffield: new		
41		
\glxstr@displayendloc: new		
143		
\glxstr@displayendlochook: new		
144		
\glxstr@displayingleloc: new		
143		
\glxstr@displaystartloc: new		
143		
\glxstr@dohyperlink: added check for		
alias field		
101		
\glxstr@deffield: new		
41		
\glxstr@entryfmt: new		
35		
\glxstr@fieldddolistloop: new		
36		
\glxstr@fieldforlistloop: new		
36		

1.13 (2017-02-07)		long-em-short-em: fixed spelling of	
\@glsdisp: removed		\glsabbrvfont	296
\@glsxtr@org@glsdisp	74	long-postshort-user: fixed spelling of	
\glsxtrsetaliasnoindex: switched to		\glsabbrvfont	320
\providecommand	97	long-postshort-user-desc: fixed	
1.14 (2017-04-18)		spelling of \glsabbrvfont	322
\@gls@link: added redefinition	77	long-short: fixed spelling of	
\@gls@noidx@getgrouptitle: new	140	\glsabbrvfont	240
\@gls@removespaces: new	144	long-short-user: fixed spelling of	
\@glsxtr@do@automake@err: new	153	\glsabbrvfont	319
\@glsxtr@org@gloautosee: new	29	footnote: fixed spelling of	
\@glsxtr@record: added third arg	7	\glsabbrvfont	246
\@glsxtr@recordsee: new	13	postfootnote: fixed spelling of	
General: added \glsadd option		\glsabbrvfont	248
theHvalue	79	1.16 (2017-06-15)	
added \glsadd option thevalue	79	\@glo@autosee: added redefinition	29
\glsdisablehyper: added redefinition	102	\@gls@noidx@getgrouptitle: fixed	
\glsenableentrycount: fixed		bug	140
assignment of \@cGls@	116	\@glsxtr@addunusedxrefs: added	
\glsenableentryunitcount: fixed		check for seealso field	61
assignment of \@cGls@	124	\@glsxtr@checkgroup: use \csuse	
\glsnavigation: new	141	instead of \csname	160
\glsxtr@org@getgrouptitle: new	141	\@glsxtr@dorecordnodefer: new	11
\glsxtr@recordsee: new	7	\@glsxtr@record@only@setup: added	
\glsxtr@writefields: added check for		check for \@gls@setupsort@none	15
automake	153	\@print@unsrt@glossary: corrected	
\glsxtrdisplayendloc: added check		misspelt command	157
for empty format	143	\@printunsrt@glossary@handler:	
\glsxtrgetgrouptitle: new	141	new	158
\glsxtrinitwrgloss: new	75	\gls@checkseeallowed: added	
\glsxtrlocationhyperlink: new	144	redefinition	29
\glsxtrsetgrouptitle: new	141	\glsxtr@writefields: added	
\glsxtrsupphypernumber: new	145	\providecommand lines	152
\ifglsxtrwrglossbefore: new	75	\glsxtrautoindex: new	204
1.15 (2017-05-10)		\glsxtrautoindexassignsort: new	204
\@glsxtr@dorecord: corrected		\glsxtrautoindexentry: new	204
premature expansion of \@glslocref	10	\glsxtrindexseealso: new	57
short-em-long-em: fixed spelling of		\glsxtrseealsolabels: new	60
\glsabbrvfont	299	\glsxtrseelist: new	57
short-long: fixed spelling of		\glsxtruseseealso: new	56
\glsabbrvfont	243	\glsxtruseseealsoformat: new	57
short-long-user: fixed spelling of		\seealso name: new	57
\glsabbrvfont	326	autoseeindex: new	18
short-postfootnote-desc: fixed		1.17 (2017-08-09)	
spelling of \glsabbrvfont	250	\@glsxtr@mark@wordseps: new	217
short-postlong-user: fixed spelling of		\@glsxtr@markwordseps: new	217
\glsabbrvfont	323	\@glsxtr@noidx@displaynumberlist:	
short-postlong-user-desc: fixed		replace hard-coded ?? with	
spelling of \glsabbrvfont	324	\glsxtrundeftag	139

\@glxtr@noidx@entrynumberlist: replace hard-coded ?? with \glxtrundeftag 140	\Glsxtrsubsequentfmt: new 236
\@glxtr@noidx@numberlistloop: replace hard-coded ?? with \glxtrundeftag 140	\glxtrsubsequentfmt: new 236
\@glxtrifhyphenstart: new 328	\Glsxtrsubsequentplfmt: new 236
General: removed some inconsistencies	\glxtrword: new 217
in the abbreviation styles 240	\glxtrwordsep: new 217
\glabbrvhyphenfont: new 329	short-hyphen-long-hyphen: new ... 338
\glabbrvonlyfont: new 343	short-hyphen-long-hyphen-desc: new 339
\glabbrvscfont: new 260	short-hyphen-postlong-hyphen: new 340
\glabbrvsmfont: new 276	short-hyphen-postlong-hyphen-desc: new 342
\glabbrvuserfont: initialised to default font 318	short-long-user-desc: corrected first forms 327
\glfirstabbrvhyphenfont: new ... 329	short-nolong-desc-noreg: new ... 255
\glfirstabbrvonlyfont: new 343	short-nolong-noreg: new 253
\glfirstabbrvscfont: new 260	long-em-noshort-em-desc-noreg: new 312
\glfirstabbrvsmfont: new 276	long-em-noshort-em-noreg: new ... 308
\glfirstlonghyphenfont: new ... 329	long-hyphen-noshort-desc-noreg: new 331
\glfirstlongonlyfont: new 343	long-hyphen-noshort-noreg: new .. 333
\glslonghyphenfont: new 329	long-hyphen-postshort-hyphen: new 334
\glslongonlyfont: new 343	long-hyphen-postshort-hyphen-desc: new 336
\glslonguserfont: initialised to default font 318	long-hyphen-short-hyphen: new ... 329
\glxtr@newabbreviation: added \glxtrorgshort and \glxtrorglong 218	long-hyphen-short-hyphen-desc: new 330
\GlsXtrDefineAcShortcuts: new 21	long-noshort-desc-noreg: new 258
\glxtrgenabbrvfmt: added check for \ifglxtrininsertinside 234	long-noshort-noreg: new 259
\glxtrhyphensuffix: new 329	long-only-short-only: new 343
\glxtrifhyphenstart: new 328	long-only-short-only-desc: new .. 345
\glxtrlonghyphen: new 333	long-short-user-desc: corrected first forms 325
\glxtrlonghyphennoshort: new ... 331	1.18 (2017-08-10)
\glxtrlonghyphenshort: new 328	stylemods: changed default value to “default” 24
\glxtrlongshortdescname: new ... 242	1.19 (2017-09-09)
\glxtronlydescname: new 345	\@glxtr@defaultnumberformat: new . 7
\glxtronlydescsort: new 345	\@glxtr@dorecord: Use
\glxtronlysuffix: new 343	\@glxtr@recordloc instead of
\glxtrparen: new 221	\@glxtr@recordloc 10
\glxtrposthyphenlong: new 340	\@glxtr@dorecordnodefer: Use
\glxtrposthyphenshort: new 334	\theglentrycounter for the
\glxtrposthyphensequent: new 334	location rather than \@glxtr@recordloc .. 11
\glxtrshortdescname: new 253	\@glxtr@record@setting: new 14
\glxtrshorthyphen: new 339	\@glxtr@record@setting@alsoindex: new 14
\glxtrshorthyphenlong: new 337	\@glxtrifhasfield: new 38
\glxtrshortlongdescname: new ... 244	
\glxtrshortlongdescsort: new ... 244	

General: added \glslink option		
theHvalue	75	
added \glslink option thevalue ...	75	
\glstr@writefields: removed		
double-quotes around \jobname ..	153	
\glstrdoautoindexname: changed		
format test	203	
\glstrhyperlink: new	101	
\glstrifhasfield: new	38	
\GlsXtrSetDefaultNumberFormat:		
new	7	
\s@glstrifhasfield: new	38	
1.20 (2017-09-11)		
\@glstrhypernameprefix: new	137	
\glsdohypertarget: added redefinition	138	
\printunrtglossaryunitsetup:		
switched from redefining		
\gloinkprefix to		
\@glstrhypernameprefix	159	
1.21 (2017-11-03)		
\@glstr@record: added check for		
default options	9	
\@glstrwrglossmark: new	27	
\glslink: changed \let to \def	102	
\glstr@checkgroup: new	160	
\glstr@defpostpunc: new	19	
\glstr@do@record@wrglossary:		
new	8	
\glstr@dosee@alsoindex@glossary:		
new	29	
\glstr@doseeglossary: new	28	
\glstr@noidx@do: removed code		
dealing with the group	161	
\glstr@record@setting@off: new ..	15	
\glstr@record@setting@only: new ..	14	
\glstr@rglstrigger@record: new ..	165	
\glstrglosseentry: new	154	
\glstrnewgls: new	162	
\glstrsetaliasnoindex: changed to		
use \glstrifhasfield instead of		
\ifglshasfield	97	
\@glstrwrglossmark: new	26	
\@rGLS: new	168	
\@rGLS@: new	168	
\@rGLSpl: new	168	
\@rGLSpl@: new	169	
\@rGls: new	167	
\@rGls@: new	167	
\@rGlspl: new	168	
\@rGlspl@: new	168	
\@rgls: new	166	
\@rgls@: new	167	
\@rglspl: new	167	
\@rglspl@: new	167	
General: adjusted mcolalttree	451	
modified index to remove hard coded		
\space	429	
modified list to remove hard coded		
\space	418	
moved conditional outside of		
\glsgroupskip	421–428	
new	454	
redefined altlistgroup to discourage		
breaks after group headings	419	
redefined altlisthypergroup to		
discourage breaks after group		
headings	420	
redefined alttreegroup to discourage		
breaks after group headings	446	
redefined alttreehypergroup to		
discourage breaks after group		
headings	447	
redefined indexgroup to discourage		
breaks after group headings	430	
redefined indexhypergroup to		
discourage breaks after group		
headings	430	
redefined listgroup to discourage		
breaks after group headings	419	
redefined listhypergroup to		
discourage breaks after group		
headings	419	
redefined mcolalttreegroup to		
discourage breaks after group		
headings	452	
redefined mcolalttreehypergroup to		
discourage breaks after group		
headings	452	
redefined mcolalttreespannav to		
discourage breaks after group		
headings	453	
redefined mcolindexgroup to		
discourage breaks after group		
headings	448	
redefined mcolindexhypergroup to		
discourage breaks after group		
headings	448	

redefined mcolindexspannav to	\glstreechildprelocation: new ...	429
discourage breaks after group	\glstreeprelocation: new	429
headings	\glstriggerrecordformat: new	166
448	\glseuseabbrvfont: new	234
redefined mcoltreegroup to	\glseuselongfont: new	234
discourage breaks after group	\glsxtr@do@alsoindex@wrglossary:	
headings	new	8
449	\glsxtr@org@@do@wrglossary: new ..	30
redefined mcoltreehypergroup to	\glsxtr@org@dohyperlink: new	100
discourage breaks after group	\glsxtr@setbookindexmark: new ...	459
headings	\glsxtrbookindexatendgroup: new ..	455
450	\glsxtrbookindexbetween: new	455
redefined	\glsxtrbookindexbookmark: new ...	455
mcoltreeonamehypergroup to	\glsxtrbookindexcols: new	454
discourage breaks after group	\glsxtrbookindexcolspread: new ..	456
headings	\glsxtrbookindexfirstmark: new ..	460
450	\glsxtrbookindexfirstmarkfmt: new	460
redefined mcoltreeonamespannav to	\glsxtrbookindexformatheader: new	455
discourage breaks after group	\glsxtrbookindexgroupskip: new ..	455
headings	\glsxtrbookindexlastmark: new ...	460
451	\glsxtrbookindexlastmarkfmt: new	460
redefined mcoltreespannav to	\glsxtrbookindexmarkentry: new ..	459
discourage breaks after group	\glsxtrbookindexname: new	454
headings	\glsxtrbookindexparentchildsep:	
450	new	455
redefined treegroup to discourage	\glsxtrbookindexparentschildsep:	
breaks after group headings	new	455
433	\glsxtrbookindexprelocation: new	454
redefined treehypergroup to	\glsxtrbookindexsubatendgroup:	
discourage breaks after group	new	455
headings	\glsxtrbookindexsubbetween: new ..	455
433	\glsxtrbookindexsubname: new	454
redefined treeonamegroup to	\glsxtrbookindexsubprelocation:	
discourage breaks after group	new	454
headings	\glsxtrbookindexsubsubatendgroup:	
434	new	455
redefined treeonamehypergroup to	\glsxtrbookindexsubsubbetween:	
discourage breaks after group	new	455
headings	\glsxtrbookindexthepage: new	459
435	\glsxtrdetoklocation: new	165
debug: new	\glxtrenablerecordcount: new ...	165
27	\glsxtrglossentry: new	154
\gglsetwidest: new	\glsxtrgroupfield: new	160
436	\Glsxtrheadname: new	352
\gl:disablehyper: added check for	\Glsxtrheadname: new	351
existence	\GlsXtrIfFieldEqStr: new	42
102	\glxtriflabelinlist: new	159
changed to use \def rather than \let	\glxtrifrecordtrigger: new	165
102		
\gl:enablehyper: changed to use \def		
rather than \let		
102		
\Glsfmtname: new		
361		
\Glsfmtname: new		
361		
\glshex: new		
371		
\glslstchildpostlocation: new ..		
418		
\glslstchildprelocation: new ...		
418		
\glslstprelocation: new		
418		
\gl:navhyperlink: patched		
100		
\gl:seeitemformat: new		
54		
\gl:showtarget: new		
28		

\glxtrindexseealso: added check that the entry exists	58	\rGLSpl: new	168
\glxtrinithyperoutside: new	76	\rGlspl: new	168
\GlsXtrLocationRecordCount: new ..	164	\rglspl: new	167
\glxtrnewgls: new	162, 163	\rGLSplformat: new	169
\glxtrnewGLSlike: new	163	\rGlsplformat: new	169
\glxtrnewglslike: new	163	\rglsplformat: new	169
\glxtrnewrgls: new	164	\s@glxtrifhasfield: switched from \ifdef to \ifundef	38
\glxtrnewrglslike: new	164	1.22 (2017-11-08)	
\glxtrprelocation: new	417, 454	\@glxtr@nopostpunc: new	136
\GlsXtrRecordCount: new	164	\@glxtr@orgprintglossary: changed explicit \let for \nopostdesc to \glxtractivatenopost	136
\glxtrrecordtriggervalue: new ..	165	\@glxtrglossentryother: new	156
\glxtrresourcefile: now disables record key	150	\glossentrynameother: new	201
\glxtrresourceinit: new	151	\glseeitemformat: switched check from regular to short	54
\GlsXtrSetRecordCountAttribute: new	165	\glxtr@setaccessdisplay: new ...	201
\Glsxtrtitlename: new	352	\glxtr@writefields: provide \glxtr@record in aux file	152
\glxtrtitlename: new	352	\glxtractivatenopost: new	136
\glxtrtitleorpdforheading: new ..	347	\glxtrbookindexprelocation: removed check for no post dot	454
\GlsXtrTotalRecordCount: new	164	\glxtrglossentryother: new	155
\glxtrwrglossmark: new	27	\glxtrnopostpunc: new	136
short-em: new	302	1.23 (2017-11-12)	
short-sc: corrected first letter uppercasing	265	\@@glxtrfmt: added check for indexing added grouping	34
short-sm: corrected first letter uppercasing	281	new	34
shortcuts: ac	23	\@glxtr@nopostpunc@postdesc: new	137
\ifglxtr@hyperoutside: new	75	\@glxtr@restore@postpunc: new ..	137
all: new	416	\@glxtrentryfmt: fixed missing label argument	35
nolong-short: new	255	\@glxtrfmt: new	34
nolong-short-em: new	304	\eglsupdatewidest: new	437
nolong-short-noreg: new	256	\gglupdatewidest: new	437
nolong-short-sc: new	267	\glsupdatewidest: new	436
nolong-short-sm: new	283	\GlsXtrDefineAbbreviationShortcuts: changed \newabbr definition to use \providecommand	21
nopostdot: new	19	\GlsXtrDefineAcShortcuts: changed \newabbr definition to use \providecommand	22
postpunc: new	19	\glxtrfmtdisplay: new	35
\printunsrtglossaryentryprocesshook: new	158	\glxtrifcustomdiscardperiod: new	212
\printunsrtglossarypredoglossary: new	158	\GlsXtrIfFieldUndef: new	40
\printunsrtglossaryskipentry: new	158	\glxtrrestorepostpunc: new	137
\rGLS: new	168	\s@@glxtrfmt: new	34
\rGls: new	167	\s@glxtrfmt: new	34
\rgls: new	166		
\rGLSformat: new	169		
\rGlsformat: new	169		
\rglsformat: new	169		

\xglupdatewidest:new	437	\glxtrcombingdiacriticIrules:	
1.24 (2017-11-14)		new	387
\gl sadd: added \@gls@setsort	80	\glxtrcombingdiacriticIVrules:	
\glxtrforcsvfield:new	37	new	388
\glxtrlocalsetgrouptitle:new ..	141	\glxtrcombingdiacriticrules:	
1.25 (2017-11-14)		new	387
\glxtrbookindexmulticolseenv:new	456	\glxtrcontrolrules:new	386
1.25 (2017-11-24)		\glxtrcurrencyrules:new	390
\glsextrapostnamehook:new	200	\glxtrdigitrules:new	413
\glxtrfootnotename:new	245	\glxtrfractionrules:new	414
\glxtrlongnoshortdescname:new ..	256	\glxtrGeneralLatinIIIrules:new	392
\glxtrlongnoshortname:new	258	\glxtrGeneralLatinIIrules:new ..	392
\glxtrlongshortname:new	240	\glxtrGeneralLatinIrules:new ..	391
\glxtrlongshortuserdescname:new	321	\glxtrGeneralLatinIVrules:new ..	393
\glxtronlyname:new	343	\glxtrGeneralLatinVIIrules:new	396
\glxtrpostlinkAddDescOnFirstUse:		\glxtrGeneralLatinVIrules:new ..	394
changed to use \glxtrparen	213	\glxtrGeneralLatinVrules:new ..	394
\glxtrpostlinkAddSymbolOnFirstUse:		\glxtrgeneralpuncIIrules:new ..	391
changed to use \glxtrparen	213	\glxtrgeneralpuncIrules:new ...	389
\glxtrshortlongname:new	242	\glxtrgeneralpuncrules:new	389
\glxtrshortlonguserdescname:new	324	\glxtrhyphenrules:new	389
\glxtrshortnolongname:new	251	\glxtrLatinA:new	397
1.26 (2018-01-05)		\glxtrLatinAA:new	399
\@glxtr@do@inc@linkcount:new ..	170	\glxtrLatinAELigature:new	399
\glslinkpresetkeys:new	76	\glxtrLatinE:new	397
\glxtr@inc@linkcount:new	76	\glxtrLatinEszettSs:new	398
\GlsXtrEnableLinkCounting:new ..	171	\glxtrLatinEszettSz:new	398
\GlsXtrIfLinkCounterDef:new	171	\glxtrLatinEth:new	398
\glxtrinclinkcounter:new	170	\glxtrLatinH:new	397
\GlsXtrLinkCounterName:new	171	\glxtrLatinI:new	397
\GlsXtrLinkCounterValue:new	170	\glxtrLatinInsularG:new	399
\GlsXtrTheLinkCounter:new	171	\glxtrLatinK:new	397
1.27 (2018-02-26)		\glxtrLatinL:new	397
\@glxtrdialecthook:new	31	\glxtrLatinLslash:new	399
General: added		\glxtrLatinM:new	397
glossaries-extra-bib2gls.sty	370	\glxtrLatinN:new	397
\Alpha:new	383	\glxtrLatinO:new	397
\Beta:new	383	\glxtrLatinOELigature:new	399
\Chi:new	384	\glxtrLatinOslash:new	399
\Digamma:new	384	\glxtrLatinP:new	398
\Epsilon:new	383	\glxtrLatinS:new	398
\Eta:new	383	\glxtrLatinSchwa:new	398
\glxtr@loaddialect:new	369	\glxtrLatinT:new	398
\glxtrBasicDigitrules:new	413	\glxtrLatinThorn:new	398
\glxtrcombingdiacriticIIrules:		\glxtrLatinWynn:new	399
new	388	\glxtrLatinX:new	398
\glxtrcombingdiacriticIrules:		\glxtrMathGreekIIrules:new	405
new	387	\glxtrMathGreekIrules:new	404

<code>\glstrMathItalicAlpha</code> : new	410	<code>\glstrUpEpsilon</code> : new	407
<code>\glstrMathItalicBeta</code> : new	410	<code>\glstrUpEta</code> : new	407
<code>\glstrMathItalicChi</code> : new	413	<code>\glstrUpGamma</code> : new	407
<code>\glstrMathItalicDelta</code> : new	410	<code>\glstrUpIota</code> : new	408
<code>\glstrMathItalicEpsilon</code> : new	410	<code>\glstrUpKappa</code> : new	408
<code>\glstrMathItalicEta</code> : new	411	<code>\glstrUpLambda</code> : new	408
<code>\glstrMathItalicGamma</code> : new	410	<code>\glstrUpMu</code> : new	408
<code>\glstrMathItalicGreekIIrules</code> :		<code>\glstrUpNu</code> : new	408
new	401	<code>\glstrUpOmega</code> : new	410
<code>\glstrMathItalicGreekIrules</code> : new	400	<code>\glstrUpOmicron</code> : new	408
<code>\glstrMathItalicIota</code> : new	411	<code>\glstrUpPhi</code> : new	409
<code>\glstrMathItalicKappa</code> : new	411	<code>\glstrUpPi</code> : new	409
<code>\glstrMathItalicLambda</code> : new	411	<code>\glstrUpPsi</code> : new	409
<code>\glstrMathItalicLowerGreekIIrules</code> :		<code>\glstrUpRho</code> : new	409
new	404	<code>\glstrUpSigma</code> : new	409
<code>\glstrMathItalicLowerGreekIrules</code> :		<code>\glstrUpTau</code> : new	409
new	403	<code>\glstrUpTheta</code> : new	407
<code>\glstrMathItalicMu</code> : new	411	<code>\glstrUpUpsilon</code> : new	409
<code>\glstrMathItalicNabla</code> : new	413	<code>\glstrUpXi</code> : new	408
<code>\glstrMathItalicNu</code> : new	411	<code>\glstrUpZeta</code> : new	407
<code>\glstrMathItalicOmega</code> : new	413	<code>\Iota</code> : new	383
<code>\glstrMathItalicOmicron</code> : new	412	<code>\Kappa</code> : new	383
<code>\glstrMathItalicPartial</code> : new	413	<code>\Mu</code> : new	383
<code>\glstrMathItalicPhi</code> : new	412	<code>\Nu</code> : new	383
<code>\glstrMathItalicPi</code> : new	412	<code>\Omicron</code> : new	384
<code>\glstrMathItalicPsi</code> : new	413	<code>\omicron</code> : new	384
<code>\glstrMathItalicRho</code> : new	412	<code>\Rho</code> : new	384
<code>\glstrMathItalicSigma</code> : new	412	<code>\Tau</code> : new	384
<code>\glstrMathItalicTau</code> : new	412	<code>\Upalpha</code> : new	384
<code>\glstrMathItalicTheta</code> : new	411	<code>\Upbeta</code> : new	384
<code>\glstrMathItalicUpperGreekIIrules</code> :		<code>\Upchi</code> : new	385
new	402	<code>\Upsilon</code> : new	384
<code>\glstrMathItalicUpperGreekIrules</code> :		<code>\Upeta</code> : new	384
new	402	<code>\Upiota</code> : new	384
<code>\glstrMathItalicUpsilon</code> : new	412	<code>\Upkappa</code> : new	384
<code>\glstrMathItalicXi</code> : new	411	<code>\Upmu</code> : new	384
<code>\glstrMathItalicZeta</code> : new	410	<code>\Upnu</code> : new	385
<code>\glstrMathUpGreekIIrules</code> : new	400	<code>\Upomicron</code> : new	385
<code>\glstrMathUpGreekIrules</code> : new	399	<code>\upomicron</code> : new	385
<code>\glstrnonprintablerules</code> : new	386	<code>\Uprho</code> : new	385
<code>\glstrprovidecommand</code> : new	372	<code>\Uptau</code> : new	385
<code>\glstrspacerules</code> : new	386	<code>\Upzeta</code> : new	384
<code>\glstrSubScriptDigitrules</code> : new	414	<code>\Zeta</code> : new	383
<code>\glstrSuperScriptDigitrules</code> : new	414		
<code>\glstrUpAlpha</code> : new	406		
<code>\glstrUpBeta</code> : new	407		
<code>\glstrUpChi</code> : new	409		
<code>\glstrUpDelta</code> : new	407		
<code>\glstrUpDigamma</code> : new	407		

1.28 (2018-03-06)	
<code>\@glstr@docdefval</code> : changed from count register to macro	17
<code>\@glstrdialecthook</code> : save and restore <code>\TrackLangRequireDialectPrefix</code>	415

\glstredfield: changed \csedef to \protected\csedef 41	\glsaddpresetkeys: new 79
\glstrlocalsetgrouptitle: changed \csedef \protected\csedef 141	\glsuserdescription: new 318
\glstrsetgrouptitle: changed \csxdef \protected\csxdef 141	\glstrabbreviationfont: new 72
1.29 (2018-04-09)	\GlsXtrDualBackLink: new 373
\@gls@removespaces: added expansion 144	\GlsXtrDualField: new 372
\@glstr@dorecord: don't suppress expansion of \@glsrecordlocref if counter isn't page 11	\GlsXtrExpandedFmt: new 76
\@glstr@wrglossary@locationhyperlink: new 26	\GLSxtrlong: added \@glstr@record 230
\glstr@inc@wrglossaryctr: new ... 25	\Glsxtrlong: added \@glstr@record 229
\glstr@wrglossarylocation: new . 372	\glstrlong: added \@glstr@record 228
\GlsXtrBibTeXEntryAliases: new .. 373	\GLSxtrlongpl: added \@glstr@record 233
\glstrfieldforlistloop: corrected argument order in \forlistcsloop 36	\Glsxtrlongpl: added \@glstr@record 233
\GlsXtrIndexCounterLink: new 372	\glstrlongpl: added \@glstr@record 232
\GlsXtrInternalLocationHyperlink: new 25	\GLSxtrshort: added \@glstr@record 228
\GlsXtrProvideBibTeXFields: new . 373	\Glsxtrshort: added \@glstr@record 227
indexcounter: new 26	\glstrshort: added \@glstr@record 226
\setentrycounter: new 144	\GLSxtrshortpl: added \@glstr@record 231
1.30 (2018-04-25)	\Glsxtrshortpl: added \@glstr@record 231
\@glstr@record: added check for post-key hook 9	\@glstr@record 230
added check for pre-key hook 9	\GlsXtrStartUnsetBuffering: new . 112
\@GLSxtr@fullpl: added \@glstr@record 226	\GlsXtrStopUnsetBuffering: new .. 112
\@GlsXtrStopUnsetBuffering: new . 113	indexcounter: added check for wrglossary counter 26
\@Glsxtr@fullpl: added \@glstr@record 225	\s@GlsXtrStopUnsetBuffering: new 113
\@glstr@dorecord: don't suppress expansion of \@glsrecordlocref .. 11	1.31 (2018-05-09)
\@glstr@full: added \@glstr@record 223	\@GlsXtrStartUnsetBuffering: new 112
\@glstr@fullpl: added \@glstr@record 225	\@gls@ifaccessattribute@set: new 180
\@glstr@glossadd@postkeys: new .. 10	\@gls@initaccesskeys: new ... 179, 188
\@glstr@glossadd@prekeys: new ... 10	\@gls@setup@default@short@access: new 181
\@glstr@glslink@postkeys: new ... 10	\@glstr@record@noglossarywarning: new 150
\@glstr@glslink@prekeys: new 10	\@glstrbuffer@nodup@unset: new . 112
\@glstr@local@textformat: new ... 75	General: added prefix key for glslink . 76
\@glstr@unset: new 111	added prefix key for printgloss .. 138
\@glstrbuffer@unset: new 112	changed \let to \def 137
\glsadd: added \glsaddpostsetkeys . 80	\glsaddeach: new 80
added \glsaddpresetkeys 80	\glsapturedgroup: new 371
\glsaddpostsetkeys: new 79	\glsdefpostdesc: new 211
	\glsdefpostlink: new 212
	\glsdefpostname: new 200

<code>\glsdohypertarget</code> : bug fix: ensure that new version is picked up	138	<code>\if@glxtrdocdefrestricted</code> : changed to allow for atom as well ...	17
<code>\glslistdesc</code> : new	418	<code>docdef</code> : atom	17
<code>\glslocalreseteach</code> : new	114	1.35 (2018-08-13)	
<code>\glslocalunseteach</code> : new	114	<code>\@gls@link@</code> : initialise post-link hook commands	74
<code>\glstreechilddesc</code> : new	432	1.36 (2018-08-18)	
<code>\glstreechildsymbol</code> : new	432	<code>\glxtrautoindexesc</code> : new	204
<code>\glstreedefaultnamefmt</code> : new	428	<code>\glxtrdisplaysupplc</code> : new	374
<code>\glstreedesc</code> : new	431	<code>\glxtrmultisupplcation</code> : new ...	374
<code>\glstreegroupheaderfmt</code> : added redefinition	429	1.37 (2018-11-30)	
<code>\glstreenamefmt</code> : added redefinition .	429	<code>\@glxtr@record</code> : added check for auto-add	9
<code>\glstreenavigationfmt</code> : added redefinition	429	<code>\@dGLS</code> : new	382
<code>\glstreenonamechilddesc</code> : new	434	<code>\@dGLSpl</code> : new	382
<code>\glstreenonamedesc</code> : new	433	<code>\@dGls</code> : new	382
<code>\glstreenonamesymbol</code> : new	433	<code>\@dGlspl</code> : new	382
<code>\glstreesymbol</code> : new	431	<code>\@dgls</code> : new	381
<code>\glxtr@newabbreviation</code> : added <code>\ExtraCustomAbbreviationFields</code>	218	<code>\@dglspl</code> : new	382
<code>\GlsXtrForUnsetBufferedList</code> : new	113	<code>\@gls@getcounterprefix</code> : new	30
<code>\GlsXtrIfFieldCmpNum</code> : new	39	<code>\@gls@longextrawidestname</code> : new ...	463
<code>\GlsXtrIfFieldEqNum</code> : new	38	<code>\@glxtr@bibgls@removespaces</code> : new	376
<code>\GlsXtrIfFieldEqXpStr</code> : new	42	<code>\@glxtr@check@bibgls@nameref</code> : new	151
<code>\GlsXtrIfFieldNonZero</code> : new	38	<code>\@glxtr@do@nameref@record</code> : new ..	12
<code>\GlsXtrIfHasNonZeroChildCount</code> : new	371	<code>\@glxtr@get@prefixedlabel</code> : new .	381
<code>\GlsXtrIfXpFieldEqXpStr</code> : new	43	<code>\@glxtr@if@record@only</code> : new	15
<code>\glxtrpostlinkAddSymbolDescOnFirstUse</code> : new	213	<code>\@glxtr@ifnum@mmode</code> : new	12
<code>\GlsXtrRecordWarning</code> : new	148	<code>\@glxtr@labelprefixes</code> : new	380
<code>\glxtrRevertToCmarks</code> : new	347	<code>\@glxtr@prefixlabellist</code> : new ...	381
<code>\GlsXtrStandaloneGlossaryType</code> : new	155	<code>\@glxtr@providenewgls</code> : new	162
<code>\GlsXtrStandaloneSubEntryItem</code> : new	155	<code>\@glxtr@record@only@setup</code> : new ..	15
<code>\s@GlsXtrStartUnsetBuffering</code> : new	112	<code>\@glxtr@record@setting@nameref</code> : new	14
1.32 (2018-05-24)		<code>\@glxtr@use@equation@counter@or</code> : new	76
<code>\GlsXtrForeignText</code> : new	43	General: new	461
<code>\GlsXtrForeignTextField</code> : new	45	page: nameref	11
<code>\GlsXtrUnknownDialectWarning</code> : new	45	<code>\dGLS</code> : new	382
1.33 (2018-07-26)		<code>\dgls</code> : new	381
<code>\ifglsused</code> : added redefinition	47	<code>\dglsdisp</code> : new	383
1.34 (2018-07-29)		<code>\dglslink</code> : new	383
<code>\gls@begindocdefs</code> : atom	62	<code>\dGLSpl</code> : new	382
<code>\GlsXtrIfUnusedOrUndefined</code> : new ..	32	<code>\glsadd</code> : added grouping	80
<code>\glxtrNoGlossaryWarning</code> : added package warning	24	ensure that <code>\glsadd</code> performs indexing	80
		<code>\gls@longextraDescAlign</code> : new	463
		<code>\gls@longextraDescFmt</code> : new	461
		<code>\gls@longextraDescNameHeader</code> : new	468

\glslongextraDescNameTabularFooter:	\glslongextraNameDescSymTabularFooter:
new 468	new 471
\glslongextraDescNameTabularHeader:	\glslongextraNameDescSymTabularHeader:
new 468	new 471
\glslongextraDescSymNameHeader:	\glslongextraNameDescTabularFooter:
new 480	new 463
\glslongextraDescSymNameTabularFooter:	\glslongextraNameDescTabularHeader:
new 480	new 463
\glslongextraDescSymNameTabularHeader:	\glslongextraNameFmt: new 461
new 480	\glslongextraNameSymDescHeader:
\glslongextraGroupHeading: new .. 463	new 474
\glslongextraHeaderFormat: new .. 463	\glslongextraNameSymDescLocationHeader:
\glslongextraLocationAlign: new . 463	new 475
\glslongextraLocationDescNameHeader:	\glslongextraNameSymDescLocationTabularFooter:
new 469	new 476
\glslongextraLocationDescNameTabularFooter:	\glslongextraNameSymDescLocationTabularHeader:
new 469	new 475
\glslongextraLocationDescNameTabularHeader:	\glslongextraNameSymDescTabularFooter:
new 469	new 474
\glslongextraLocationDescSymNameHeader:	\glslongextraNameSymDescTabularHeader:
new 481	new 474
\glslongextraLocationDescSymNameTabularFooter:	\glslongextraSetDescWidth: new .. 464
new 482	\glslongextraSetWidest: new 463
\glslongextraLocationDescSymNameTabularHeader:	\glslongextraSubDescFmt: new 462
new 482	\glslongextraSubLocationFmt: new 462
\glslongextraLocationFmt: new ... 462	\glslongextraSubNameFmt: new 462
\glslongextraLocationSymDescNameHeader:	\glslongextraSubSymbolFmt: new .. 462
new 478	\glslongextraSymbolAlign: new ... 463
\glslongextraLocationSymDescNameTabularFooter:	\glslongextraSymbolFmt: new 461
new 479	\glslongextraSymDescNameHeader:
\glslongextraLocationSymDescNameTabularHeader:	new 477
new 478	\glslongextraSymDescNameTabularFooter:
\glslongextraLocSetDescWidth: new 465	new 477
\glslongextraNameAlign: new 463	\glslongextraSymDescNameTabularHeader:
\glslongextraNameDescHeader: new 463	new 477
\glslongextraNameDescLocationHeader:	\glslongextraSymLocSetDescWidth:
new 466	new 465
\glslongextraNameDescLocationTabularFooter:	\glslongextraSymSetDescWidth: new 464
new 467	\glslongextraTabularVAlign: new . 465
\glslongextraNameDescLocationTabularHeader:	\glslongextraUpdateWidest: new .. 464
new 466	\glslongextraUpdateWidestChild:
\glslongextraNameDescSymHeader:	new 464
new 471	\glsrenewcommand: new 372
\glslongextraNameDescSymLocationHeader:	\glsseeitemformat: removed reference
new 472	to \glslabel 54
\glslongextraNameDescSymLocationTabularFooter:	\glsxtr@dblfloat: new 18
new 472	\glsxtr@do@autoadd: new 76
\glslongextraNameDescSymLocationTabularHeader:	\glsxtr@float: new 18
new 472	\glsxtr@record@nameref: new 154

\glstr@renewcommand: new	372	1.38 (2018-12-01)	
\glstr@writefields: provide			\glslongextraNameFmt: bug fix:
\glstr@record@nameref in aux			removed double param
file	152		all: added glossary-longextra
\glstraddlabelprefix: new	380	1.39 (2019-03-22)	
\GlsXtrAutoAddOnFormat: new	76		\@GlsXtrIfFieldCmpNum: new
\glstrclearlabelprefixes: new	380		\@GlsXtrIfFieldEqNum: new
\glstrdisplaylocnameref: new	374		\@GlsXtrIfFieldEqStr: new
\glstrfmtexternalnameref: new	377		\@GlsXtrIfFieldEqXpStr: new
\glstrfmtinternalnameref: new	376		\@GlsXtrIfFieldNonZero: new
\GLSXRhiername: new	56		\@GlsXtrIfXpFieldEqXpStr: new
\GLSxtrhiername: new	56		\@glstr@removespaces: changed \x to
\GlsXtrhiername: new	55		\@glo@tmp
\Glsxtrhiername: new	55		\@glstr@dorecord: added protection
\glstrhiername: new	54		for fragile commands
\glstrhiernamesep: new	56		General: added label key for
\glstridentifyglslike: new	162		printgloss
\glstrifinlabelprefixlist: new	381		\glstrbookindexlocation: new
\GlsXtrLocationField: new	160		\glstrbookindexsublocation: new
\glstrnameloclink: new	375		\glstrentryparentname: new
\glstrnamerefink: new	375		\GlsXtrIfFieldCmpNum: added starred
\glstrprependlabelprefix: new	380		version
\GlsXtrSetAltModifier: write modifier			\GlsXtrIfFieldEqNum: added starred
to aux	100		version
\glstrSetWidest: new	377		\GlsXtrIfFieldEqStr: added starred
\glstrSetWidestFallback: new	379		form
\GlsXtrStandaloneEntryName: new	155		\GlsXtrIfFieldEqXpStr: added starred
\GlsXtrStandaloneEntryOther: new	156		form
\GLSxtrusefield: new	40		\GlsXtrIfFieldNonZero: added starred
\Glsxtrusefield: fixed internal			version
command and added check for			\GlsXtrIfXpFieldEqXpStr: added
\texorpdfstring	40		starred form
\ifGlsLongExtraUseTabular: new	465		\glstrsetglossarylabel: new
floats: new	18		\glstrshortdescname: corrected to
long-desc-name: new	468		show long form as advertised in the
long-desc-sym-name: new	480		manual
long-loc-desc-name: new	470		short-desc: corrected to omit
long-loc-desc-sym-name: new	482		description key as advertised in the
long-loc-sym-desc-name: new	479		manual
long-name-desc: new	465		short-em-desc: bug fix: omit description
long-name-desc-loc: new	467		key as advertised in the manual
long-name-desc-sym: new	471		short-sc-desc: bug fix: omit description
long-name-desc-sym-loc: new	473		key as advertised in the manual
long-name-sym-desc: new	474		short-sm-desc: corrected to omit
long-name-sym-desc-loc: new	476		description key as advertised in the
long-sym-desc-name: new	477		manual
equations: new	18		\s@GlsXtrIfFieldCmpNum: new
			\s@GlsXtrIfFieldEqNum: new
			\s@GlsXtrIfFieldEqStr: new

\s@GlsXtrIfFieldEqXpStr: new	43	1.41 (2019-04-09)	General: changed \thisgrptitle to	
\s@GlsXtrIfXpFieldEqXpStr: new	43		\glxtrcurrentgrptitle	459
1.4.2 (??)			\glslistgroupskip: new	418
\@glossentrysymbol: new	208		\glstopicAssignSubIndent: moved	
\glstentrypdfsymbol: new	208		\par from \glstopicSubItem	486
1.40 (2019-03-22)			\glstopicSubItem: added check for	
General: new	484		description	487
\glstopicAssignSubIndent: new	486		moved \par to	
\glstopicAssignWidest: new	486		\glstopicAssignSubIndent	487
\glstopicCols: new	488		\glstopicSubLoc: moved \space to	
\glstopicColsEnv: new	488		\glstopicSubPreLocSep	488
\glstopicDesc: new	486		\glstopicSubPreLocSep: new	488
\glstopicGroupHeading: new	485		\glstreeChildDescLoc: new	432
\glstopicInit: new	486		\glstreeDescLoc: new	431
\glstopicItem: new	485		\glstreegroupskip: new	429
\glstopicLoc: new	486		\glstreePreHeader: new	429
\glstopicMarker: new	485		\glxtralttreeSymbolDescLocation:	
\glstopicMidSkip: new	487		added check for description	435
\glstopicName: new	485		topic: added penalty if no description	484
\glstopicParIndent: new	486		topicmcols: added penalty if no	
\glstopicPostSkip: new	487		description	488
\glstopicPreSkip: new	487	1.42 (2020-02-03)		
\glstopicSubIndent: new	486		\@@glxtr@record: moved label	
\glstopicSubItem: new	487		definition outside of conditional	9
\glstopicSubItemBox: new	487		\@ACRfull: added redefinition	93
\glstopicSubItemSep: new	487		\@ACRfullpl: added redefinition	94
\glstopicSubLoc: new	488		\@Acrfull: added redefinition	93
\glstopicSubNameFont: new	488		\@Acrfullpl: added redefinition	93
\glstopicTitleFont: new	485		\@GlsXtrIfFieldValueInCsvList:	
\glstopicwidest: new	486		new	37
all: added glossary-topic	416		\@acrfull: added redefinition	93
topic: new	484		\@acrfullpl: added redefinition	93
topicmcols: new	488		\@domakeglossaries: provided	
1.40 (2019-03-31)			definition for \@domakeglossaries	131
\glstfirstabbrvdefaultfont: changed			\@glstassign@actual: new	181
definition from \glstabbrvfont to			\@glstentry@field: redefined	47
\glstabbrvdefaultfont for			\@glstsetup@default@access: added	
consistency	222		\glstdefaultshortaccess	181
\GlsXtrDefaultResourceOptions:			\@glstsetup@default@short@access:	
new	150		renamed to	
long-hyphen-noshort-noreg:			\@glstsetup@default@access	181
corrected formatting commands	333		\@glstlink: switched from	
\printunsrtabbreviations: new	371		\glstdohyperlink to	
\printunsrtacronyms: new	370		\glstxtrdohyperlink	102
\printunsrtindex: new	370		\@glstxtr@abbrlists: new	128
\printunsrtnumbers: new	371		\@glstxtr@acronymlists: new	128
\printunsrtsymbols: new	370		\@glstxtr@addabbreviationlist: new	129
			\@glstxtr@base@acrcmd: new	88

\@glxstr@doloadprefix: new	24	removed \relax and updated	
\@glxstr@org@addtoacronymlists:		\@gls@short instead of	
new	128	\glsshorttok	219
\@glxstr@org@setacronymlists: new	128	replaced explicit \spacefactor with	
\@glxstr@entryfmt: added \glslabel		\@	219
and scope	35	\glxstr@writefields: added check for	
General: added \@afterheading	448	order=letter	153
debug: showaccsupp	27	\glxstrAccSuppAbbrSetFirstLongAttrs:	
\forall abbreviations: new ...	128	new	184, 188
\forall acronyms: new	129	\glxstrAccSuppAbbrSetNameLongAttrs:	
\glsdefaultshortaccess: new	180	new	185, 188
\glsdisplaynumberlist: added	371	\glxstrAccSuppAbbrSetNameShortAttrs:	
\glsenablehyper: switched from		new	185, 188
\glsdohyperlink to		\glxstrAccSuppAbbrSetNoLongAttrs:	
\glxstrdohyperlink	102	new	184, 188
\glstrynumberlist: added	371	\glxstrAccSuppAbbrSetTextShortAttrs:	
\GLSfmtfirst: new	364	new	184, 188
\GLSfmtfirstpl: new	365	\glxstralttreeSymbolDescLocation:	
\GLSfmtfull: new	368	switched to using \glstreeDescLoc	435
\Glsfmtfull: switched pdf case to use		\glxstrassignactualsetup: new ...	180
\glspdffmtfull	368	\glxstrbookindexbookmarkprefix:	
\glspdffmtfull: switched pdf case to use		new	456
\glspdffmtfull	367	\GlsXtrDiscardUnsetBuffering: new	113
\GLSfmtfullpl: new	369	\glxstrdohyperlink: new (was former	
\Glsfmtfullpl: switched pdf case to use		redefinition of \glsdohyperlink) .	101
\glspdffmtfullpl	368	\glxstreuationlocfmt: new	375
\glspdffmtfullpl: switched pdf case to use		\glxstrfieldformatcsvlist: new ...	37
\glspdffmtfullpl	368	\glxstrfieldformatlist: new	36
\GLSfmtlong: new	366	\glxstrfootnotedescname: new ...	247
\GLSfmtlongpl: new	367	\glxstrfootnotedescsort: new ...	247
\GLSfmtname: new	362	\GLSXtrhiername: switched to using	
\GLSfmtplural: new	363	\GLSfmtttext and \GLSfmtname ...	56
\GLSfmtttext: new	363	\GLSXtrhiername: switched to using	
\glspdffmtfull: new	367	\glsfmmttext, \glsfmtname,	
\glspdffmtfullpl: new	367	\GLSfmtttext and \GLSfmtname ...	56
\glseeitemformat: switched to using		\GlsXtrhiername: switched to using	
\glsfmmttext and \glsfmtname ...	54	\glsfmmttext and \glsfmtname ...	55
\glsshowtarget: added check for		\GlsXtrhiername: switched to using	
\glsshowtargetouter	28	\glsfmmttext and \glsfmtname ...	55
\glstreeChildDescLoc: added		\glxstrhiername: switched to using	
\glstreeNoDescSymbolPreLocation		\glsfmmttext and \glsfmtname ...	54
.....	432	\GlsXtrIfFieldValueInCsvList: new	37
\glstreegroupheaderskip: new ...	429	\glxstrpdfentryfmt: new	35
\glstreeNoDescSymbolPreLocation:		\glxstrprovideaccsuppcmd: new ...	184
new	431	\glxstrscsuffix: added \protect ..	260
\glxstr@newabbreviation: moved		\GlsXtrSetAltModifier: added check	100
apply abbreviation style to after		\GLSXtrtitlefirst: new	355
category key has been obtained ...	218	\GLSXtrtitlefirstplural: new ...	356
		\GLSXtrtitlefull: new	359

\GLSxtrtitlefullpl: new	360	moved \protect inside	
\GLSxtrtitlelong: new	357	\glxtrscsuffix	271
\GLSxtrtitlelongpl: new	358	short-sc-footnote-desc: new	273
\GLSxtrtitlename: new	352	short-sc-long: added missing text key	262
\GLSxtrtitleplural: new	354	moved \protect inside	
\GLSxtrtitleshort: new	351	\glxtrscsuffix	262
\GLSxtrtitleshortpl: new	351	short-sc-postfootnote: added	
\GLSxtrtitletext: new	353	missing text key	273
\glxtrusealias: new	57	moved \protect inside	
short-em: removed \protect from		\glxtrscsuffix	274
\glxtremsuffix	301	short-sc-postfootnote-desc: new .	275
short-em-desc: removed \protect		short-sm: removed \protect from	
from \glxtremsuffix	303	\glxtrsmsuffix	281
short-em-footnote: added missing text		short-sm-desc: removed \protect	
key	312	from \glxtrsmsuffix	282
removed \protect from		short-sm-footnote: added missing text	
\glxtremsuffix	313	key	287
short-em-footnote-desc: new	314	removed \protect from	
short-em-long: added missing text key	297	\glxtrsmsuffix	288
removed \protect from		short-sm-footnote-desc: new	289
\glxtremsuffix	298	short-sm-long: added missing text key	278
short-em-long-em: added missing text		removed \protect from	
key	299	\glxtrsmsuffix	279
removed \protect from		short-sm-postfootnote: added	
\glxtremsuffix	300	missing text key	290
short-em-postfootnote: added		removed \protect from	
missing text key	315	\glxtrsmsuffix	291
removed \protect from		short-sm-postfootnote-desc: new .	292
\glxtremsuffix	316	\makeglossaries: added	
short-em-postfootnote-desc: new .	317	\@domakeglossaries	131
short-footnote-desc: new	247	let \@makeglossary to \@gobble	
short-hyphen-long-hyphen: added		instead of \relax	132
missing text key	338	removed redefinition of	
short-hyphen-postlong-hyphen:		\makeglossary	132
added missing text key	340	\makenoidxglossaries: added	
short-long: added missing text key ...	243	\@domakeglossaries	63
short-long-user: added missing text		long-em-noshort-em: removed	
key	326	\protect from \glxtremsuffix .	307
short-postfootnote-desc: added		long-em-noshort-em-desc: removed	
missing text key	250	\protect from \glxtremsuffix .	310
new	250	long-em-short-em: added missing text	
short-postlong-user: added missing		key	295
text key	322	removed \protect from	
short-sc: moved \protect inside		\glxtremsuffix	296
\glxtrscsuffix	264	long-hyphen-noshort-desc-noreg:	
short-sc-desc: moved \protect inside		added missing text key	331
\glxtrscsuffix	266	long-hyphen-postshort-hyphen:	
short-sc-footnote: added missing text		added missing text key	334
key	271		

long-hyphen-short-hyphen: added missing text key	329	removed \protect from \glxxtremsuffix	294
long-noshort-em: removed \protect from \glxxtremsuffix	305	long-short-sc: added missing text key moved \protect inside \glxxtremsuffix	260
long-noshort-em-desc: removed \protect from \glxxtremsuffix .	309	long-short-sm: added missing text key removed \protect from \glxxtremsuffix	277
long-noshort-sc: moved \protect inside \glxxtremsuffix	268	long-short-user: added missing text key	319
long-noshort-sc-desc: moved \protect inside \glxxtremsuffix	269	footnote: added missing text key	246
long-noshort-sm: removed \protect from \glxxtremsuffix	284	footnote-desc: new	248
long-noshort-sm-desc: removed \protect from \glxxtremsuffix .	286	postfootnote: added missing text key .	248
long-only-short-only: added missing text key	343	prefix: new	24
removed \protect from \glxxtremsuffix	344	\RestoreAcronyms: added display style	130
long-postshort-user: added missing text key	320	\s@GlsXtrIfFieldValueInCsvList: new	38
long-short: added missing text key ...	240	\seealsoname: add check for \alsoname	57
long-short-em: added missing text key	293	1.42 (?) postfootnote-desc: new	251
		1.43 (2020-02-28) \@glxxtrentryfmt: changed \def to \edef to avoid infinite recursion ...	35

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
\\$	371
\,	56
\.	19, 212, 213, 428
\@	62, 150, 180, 219
\@@cGLS@	116, 124
\@@cGLSpl@	116, 124
\@@cGls@	116, 124
\@@cGlspl@	116, 124
\@@cgls@	116, 124
\@@cglspl@	116, 124
\@do@wrglossary	8, 10, 133
\@do@wrglossary	13, 15, 16, 30, 80, 97
\@glo@assign@sortkey	139
\@glo@list	6
\@glo@type	157
\@glossarysec	455
\@glossaryseclabel	138
\@gls@expand@field	33
\@glslocalreset	113
\@glslocalunset	113
\@glsreset	113
\@glsunset	112
\@glsxtr@autoindex@escspch	205–207
\@glsxtr@base@acrcmd@warn	88, 129
\@glsxtr@checkspch	204, 205, 207
\@glsxtr@disabledflycommand	67, 68
\@glsxtr@org@postdescription	136, 137
\@glsxtr@record	15, 16
\@glsxtr@recordcounter	15, 16, 154
\@glsxtrfmt	34
\@glsxtrp	107
\@glsxtrp@postloctag	70
\@glsxtrp@preloctag	70
\@glsxtrwrglossmark	8, 9, 13, 29, 30, 58, 63, 133
\@@newglossaryentry@defcounters	115
\@@newglossaryentry@defunitcounters	123
\@par	435
\@ACRlong	103
\@ACRlongpl	103
\@ACRshort	103
\@ACRshortpl	103
\@Acrlong	103
\@Acrlongpl	103
\@Acrshort	103
\@Acrshortpl	103
\@GLS@	103, 118, 119, 168, 382
\@GLSdesc@	85
\@GLSpl@	103, 119, 120, 169, 382
\@GLSplural@	104
\@GLSsymbol@	86
\@GLSxtr@full	224
\@GLSxtr@fullpl	226
\@GLSxtr@p@acrlong@	103
\@GLSxtr@p@acrlongpl@	103
\@GLSxtr@p@acrshort@	103
\@GLSxtr@p@acrshortpl@	103
\@GLSxtr@p@long@	103
\@GLSxtr@p@longpl@	103
\@GLSxtr@p@plural@	103
\@GLSxtr@p@short@	103
\@GLSxtr@p@shortpl@	103
\@GLSxtr@p@text@	103
\@GLSxtrlong	103, 229
\@GLSxtrlongpl	103, 233
\@GLSxtrp	110, 111
\@GLSxtrshort	103, 228
\@GLSxtrshortpl	103, 231
\@Gls@	103, 118, 119, 167, 382
\@Gls@acrentryname	127
\@Gls@entry@field	40, 94, 109, 110, 202
\@Gls@entryname	127

\@GlsXtrEnableOnTheFly	64, 65	\@cGls@	116, 124
\@GlsXtrIfFieldCmpNum	38, 39	\@cGlspl@	116, 124
\@GlsXtrIfFieldEqNum	39	\@cgl@	116, 124
\@GlsXtrIfFieldEqStr	42	\@cglpl@	116, 124
\@GlsXtrIfFieldEqXpStr	42	\@currentlabelname	138
\@GlsXtrIfFieldNonZero	38	\@dGLS	382
\@GlsXtrIfFieldValueInCsvList	37	\@dGLSpl	382
\@GlsXtrIfXpFieldEqXpStr	43	\@dGls	382
\@GlsXtrStartUnsetBuffering	112	\@dGlspl	382
\@GlsXtrStopUnsetBuffering	112	\@dblfloat	18
\@Glspl@	103, 118, 119, 168, 382	\@dgls	381
\@Glsplural@	104	\@dglspl	382
\@Gls@	103	\@disable@onlypremakeg	132
\@Glsxtr	66, 67	\@do@auxoutstuff	145, 146
\@Glsxtr@full	224	\@do@gl@getcounterprefix	11, 12
\@Glsxtr@fullpl	225	\@do@glsee	59, 60
\@Glsxtr@p@acrlong@	103	\@do@newglossaryentry	127, 128, 220
\@Glsxtr@p@acrlongpl@	103	\@do@seeglossary	15, 16, 28, 63, 132
\@Glsxtr@p@acrshort@	103	\@do@wrglossary	79, 166
\@Glsxtr@p@acrshortpl@	103	\@domakeglossaries	63, 131
\@Glsxtr@p@long@	103	\@dtl@formatlist@handler	36
\@Glsxtr@p@longpl@	103	\@dtl@formatlist@itemsep	36
\@Glsxtr@p@plural@	103	\@dtl@formatlist@lastitem	36
\@Glsxtr@p@short@	103	\@dtl@formatlist@prelastitem	36
\@Glsxtr@p@shortpl@	103	\@dtl@formatlist@prelastitemsep	36
\@Glsxtr@p@text@	103	\@dtlformatlist	37
\@Glsxtrlong	103, 229	\@empty	31,
\@Glsxtrlongpl	103, 233		81, 89–93, 129, 137, 144, 204, 205, 223–233
\@Glsxtrp	109, 110	\@end@glxtr@addunused	61
\@Glsxtrpl	67	\@end@glxtr@gettype	135, 138
\@Glsxtrshort	103, 227	\@end@glxtr@usesee	54
\@Glsxtrshortpl	103, 231	\@end@glxtrifhyphenstart	328
\@acrlong	103	\@endfortrue	37, 201, 237, 381
\@acrlongpl	103	\@firstofone	81, 157,
\@acrshort	103		180, 181, 195, 196, 203, 208, 209, 375, 376
\@acrshortpl	103	\@firstofthree	74,
\@addtoacronymlists	127–130		81, 89–92, 99, 223, 225, 227, 228, 230, 232
\@addtoreset	170	\@firstoftwo	82–86, 90, 92, 93, 96, 99, 130,
\@afterheading	419,		201, 214, 215, 224–226, 230–233, 347, 348
	420, 430, 431, 433–435, 448–451, 459, 485	\@float	18
\@alt@gl@hyp@opt	99	\@for	6, 25,
\@auxout			37, 61, 80, 114, 115, 126, 129, 130, 132,
	11–13, 63, 71, 100, 116, 117, 125, 132,		135, 142, 157, 165, 171, 194, 201, 209, 381
	133, 145, 146, 150, 152–154, 162, 381, 459	\@glo@alias	59, 60
\@bibgl@restoreat	150	\@glo@assign@sortkey	134
\@cGLS	119	\@glo@autosee	29
\@cGLS@	116, 119, 124	\@glo@autoseehook	59
\@cGLSpl	119	\@glo@category	121
\@cGLSpl@	116, 120, 124	\@glo@check@sortallowed	135

<code>\@glo@counterprefix</code> ...	11, 12, 31, 144, 376	<code>\@gls@default@longpl</code>	218–220
<code>\@glo@countunit</code>	121	<code>\@gls@deffile</code>	62
<code>\@glo@default@sorttype</code>	135	<code>\@gls@doautomake</code>	135, 153
<code>\@glo@desc</code>	48	<code>\@gls@doautomake@err</code>	153
<code>\@glo@descplural</code>	48	<code>\@gls@enablesavenonumberlist</code>	62
<code>\@glo@group</code>	14	<code>\@gls@encapchar</code>	205
<code>\@glo@label</code> 14, 33, 53, 59–61, 94, 102, 438–444		<code>\@gls@entry@count</code>	116, 117
<code>\@glo@location</code>	14	<code>\@gls@entry@field</code>	
<code>\@glo@loclist</code>	14	33, 40, 59, 94, 108–111, 115, 155, 156
<code>\@glo@name</code>	204	<code>\@gls@entry@unitcount</code>	124, 125
<code>\@glo@no@assign@sortkey</code>	138	<code>\@gls@field@font</code>	81–88
<code>\@glo@parent</code>	439, 440	<code>\@gls@field@link</code>	81–88, 94, 95
<code>\@glo@see</code>	53, 54, 56, 57, 60, 61	<code>\@gls@firstaccess</code>	179, 183
<code>\@glo@seealso</code>	59, 60	<code>\@gls@firstpluralaccess</code>	179, 183
<code>\@glo@sort</code>	204	<code>\@gls@get@counterprefix</code>	31
<code>\@glo@sorttype</code>	135, 142	<code>\@gls@getcounterprefix</code>	11
<code>\@glo@text</code>	74	<code>\@gls@getgrouptitle</code>	141, 157
<code>\@glo@thislettergrp</code>	160	<code>\@gls@grptitle</code>	100, 142
<code>\@glo@thisvalue</code>	318	<code>\@gls@hyp@opt</code>	94, 95,
<code>\@glo@tmp</code>	31, 33, 57, 94, 144	99, 119, 163, 166–168, 223–233, 381, 382	
<code>\@glo@type</code> ...	61, 100, 127, 129, 132, 135,	<code>\@gls@hyp@opt@cs</code>	99
136, 138, 142, 143, 145, 146, 149, 150, 157		<code>\@gls@ifaccessattribute@set</code>	182
<code>\@glo@types</code>	192, 438–444	<code>\@gls@ifinlist</code>	159
<code>\@glossary@default@style</code>	68, 135, 453	<code>\@gls@increment@currcount</code>	115
<code>\@glossarystyle</code>	135, 136	<code>\@gls@increment@currunitcount</code>	124
<code>\@glossentrysymbol</code>	208	<code>\@gls@initaccesskeys</code>	218
<code>\@gls@</code>	103, 117, 119, 167, 381	<code>\@gls@keymap</code>	14, 33, 54, 59, 94, 152, 201
<code>\@gls@@automake@immediate</code>	132	<code>\@gls@label</code>	8,
<code>\@gls@@link</code>	74	9, 11, 12, 63, 98, 99, 132, 133, 154, 237	
<code>\@gls@ReturnAfterFi</code>	144, 376	<code>\@gls@levelchar</code>	205
<code>\@gls@actualchar</code>	204	<code>\@gls@link</code>	34, 73, 74, 89–93, 223–234
<code>\@gls@actuallylong</code>	181, 182	<code>\@gls@link@checkfirsthyper</code>	74, 130
<code>\@gls@actuallylongpl</code>	181, 182	<code>\@gls@link@label</code>	77, 166
<code>\@gls@actualshort</code>	181, 182	<code>\@gls@link@nocheckfirsthyper</code>	
<code>\@gls@actualshortpl</code>	181, 182	73, 89–93, 223–233
<code>\@gls@adjustmode</code>	80	<code>\@gls@link@opts</code>	77
<code>\@gls@alt@hyp@opt</code>	99	<code>\@gls@list</code>	142
<code>\@gls@alt@hyp@opt@char</code>	99, 100	<code>\@gls@local@increment@currcount</code> ...	116
<code>\@gls@alt@hyp@opt@keys</code>	99, 100	<code>\@gls@local@increment@currunitcount</code> 124	
<code>\@gls@assign@actual</code>	181	<code>\@gls@location</code>	161
<code>\@gls@automake</code>	135	<code>\@gls@loclist</code>	139, 140, 161
<code>\@gls@between</code>	142	<code>\@gls@long</code>	219
<code>\@gls@checkedmkidx</code>	204, 205, 207	<code>\@gls@longaccess</code>	179, 180, 182
<code>\@gls@checkmkidxchars</code>	58, 204	<code>\@gls@longaccesspl</code>	179, 180, 182
<code>\@gls@codepage</code>	146	<code>\@gls@longpl</code>	181, 216, 218–220
<code>\@gls@counter</code>		<code>\@gls@map</code>	201
.....	9, 11, 12, 26, 31, 76, 77, 80, 97, 166	<code>\@gls@nameaccess</code>	179, 183
<code>\@gls@currentlettergroup</code> ...	142, 157, 160	<code>\@gls@nohyperlist</code>	49, 51
<code>\@gls@declareoption</code>	5	<code>\@gls@noidx@do</code>	142

\@gls@noidx@getgrouptitle	157	\@gls@longextra@begintab	465–479, 481–483
\@gls@noidx@nosanitizesort	134	\@gls@longextrawidestname	463, 464
\@gls@noidx@sanitizesort	134	\@gls@nextpages	136
\@gls@noidx@loclist@finalsep	139	\@gls@nonextpages	136
\@gls@noidx@loclist@prev	139	\@gls@numberformat	
\@gls@noidx@loclist@sep	139	...	9, 11, 12, 77, 80, 97, 166, 200, 203, 376
\@gls@noref@warn	133, 143	\@gls@order	132
\@gls@org@glsnoidx@displayloc ..	139, 140	\@gls@pl@	103, 118, 119, 167, 382
\@gls@org@gls@seeformat	139, 140	\@gls@plural@	104
\@gls@org@longpl	220	\@gls@punc@token	215
\@gls@org@shortpl	220	\@gls@recordlocref	11
\@gls@pluralaccess	179, 183	\@gls@showtarget	101
\@gls@preglossaryhook	136, 209	\@gls@style@altlist	419
\@gls@prevlevel	445–447, 451–453	\@gls@style@altlistgroup	419
\@gls@quotechar	204	\@gls@style@altlisthypergroup	420
\@gls@reference	63, 132, 133	\@gls@style@alttree	435
\@gls@restoreat	62	\@gls@style@alttreegroup	447
\@gls@saveentrycounter	15, 16, 30, 78, 80, 166	\@gls@style@alttreehypergroup	447
\@gls@see@noindex	29, 150, 151	\@gls@style@index	429
\@gls@setdefault@glslink@opts	9, 34, 77, 98	\@gls@style@indexgroup	430
\@gls@setsort	78, 80	\@gls@style@indexhypergroup	430
\@gls@setup@default@access	220	\@gls@style@inline	428
\@gls@setupsort@none	15	\@gls@style@list	418
\@gls@short	219	\@gls@style@listdotted	417
\@gls@shortaccess	179–183	\@gls@style@listgroup	419
\@gls@shortaccesspl	179, 180, 182–184	\@gls@style@listhypergroup	419
\@gls@shortpl	181, 216, 219, 220	\@gls@style@mcolalttree	451
\@gls@sort	160	\@gls@style@mcolalttreegroup	452
\@gls@textaccess	179, 183	\@gls@style@mcolalttreehypergroup ..	452
\@gls@thisHloc	31	\@gls@style@mcolalttreespannav	453
\@gls@thislabel	80, 114, 381–383	\@gls@style@mcolindexgroup	448
\@gls@thisloc	31	\@gls@style@mcolindexhypergroup	448
\@gls@thisval	201	\@gls@style@mcolindexspannav	448
\@gls@tmp	43, 142, 181	\@gls@style@mcoltreegroup	449
\@gls@tmpb	207	\@gls@style@mcoltreehypergroup	449
\@gls@type	129, 130, 132, 133, 135, 237, 438–444	\@gls@style@mcoltreenamegroup	450
\@gls@write@entrycounts	116	\@gls@style@mcoltreenamehypergroup	450
\@gls@write@entryunitcounts	124	\@gls@style@mcoltreenameespannav ..	451
\@gls@write@entryunitcounts@do	125	\@gls@style@mcoltreespannav	450
\@gls@writedef	62, 63	\@gls@style@tree	431
\@gls@xref	13, 58	\@gls@style@treegroup	433
\@gls@abbrv@current@abbreviation	218, 234	\@gls@style@treehypergroup	433
\@gls@acronymlists	127–130	\@gls@style@treename	433
\@gls@doifexistsorwarn ...	17, 196, 198–201	\@gls@style@treenamegroup	434
\@gls@entry	62, 116, 117, 125	\@gls@style@treenamehypergroup ...	435
\@gls@link	79, 100, 102	\@gls@target	102, 137, 138
\@gls@localreset	114	\@gls@text@	103
\@gls@localunset	113, 114	\@gls@unset	112, 113
		\@gls@widestname	437, 445

<code>\@glsxtr</code>	65, 67	<code>\@glsxtr@current@style</code>	68, 453
<code>\@glsxtr@do@wrglossary</code>	133	<code>\@glsxtr@currentunitcount</code> ..	121, 122, 124
<code>\@glsxtr@abbreviationsdef</code>	20, 29, 30	<code>\@glsxtr@currunitcount</code>	123, 125
<code>\@glsxtr@abbrlists</code>	129	<code>\@glsxtr@debugnr</code>	27
<code>\@glsxtr@accessdisplay</code>	201, 202	<code>\@glsxtr@debugval</code>	27
<code>\@glsxtr@acronymlists</code>	129, 130	<code>\@glsxtr@declareoption</code> ...	5, 18–20, 24, 26
<code>\@glsxtr@activate@initialtagging</code> ..		<code>\@glsxtr@defaultnoglossarywarning</code> ..	24
.....	209, 210	<code>\@glsxtr@defaultnumberformat</code>	
<code>\@glsxtr@addabbreviationlist</code>	221	7, 9, 77, 80, 97, 200, 203
<code>\@glsxtr@addunitcounter</code>	121	<code>\@glsxtr@defpostpunc</code>	19, 20, 27
<code>\@glsxtr@addunused</code>	61	<code>\@glsxtr@deprecated@abbrstyle</code>	
<code>\@glsxtr@addunusedxrefs</code>	61, 62	269, 271, 272,
<code>\@glsxtr@altmodifier</code>	100	275, 286, 287, 289, 292, 306, 310, 314, 316	
<code>\@glsxtr@attrval</code>	78, 195–204, 208	<code>\@glsxtr@dialect</code>	44, 45
<code>\@glsxtr@autoindex@at</code>	204, 205	<code>\@glsxtr@disabledflycommand</code>	67
<code>\@glsxtr@autoindex@doextra@esc</code> ...	204	<code>\@glsxtr@display@loc</code>	143
<code>\@glsxtr@autoindex@encap</code>	204–206	<code>\@glsxtr@do@windex</code>	98, 99
<code>\@glsxtr@autoindex@esc</code>	204, 206, 207	<code>\@glsxtr@do@autoadd</code>	77
<code>\@glsxtr@autoindex@escat</code>	205	<code>\@glsxtr@do@gl:disablehyperinlist</code> ..	96
<code>\@glsxtr@autoindex@escencap</code> ...	205, 206	<code>\@glsxtr@do@inc@linkcount</code>	171
<code>\@glsxtr@autoindex@esclevel</code> ...	205, 206	<code>\@glsxtr@do@nameref@record</code>	11, 12
<code>\@glsxtr@autoindex@escquote</code> ...	204, 206	<code>\@glsxtr@do@record@wrglossary</code> ...	8, 15
<code>\@glsxtr@autoindex@level</code>	205, 206	<code>\@glsxtr@do@redef@forgl:sentries</code>	7
<code>\@glsxtr@autoindex@setname</code>	203	<code>\@glsxtr@do@style</code>	25, 370
<code>\@glsxtr@autoindex@crossrefs</code> 15, 17, 54, 59		<code>\@glsxtr@do@titlecaps@warn</code>	
<code>\@glsxtr@autoseeindexfalse</code>	15	195–198, 202, 210
<code>\@glsxtr@autoseeindextrue</code>	18	<code>\@glsxtr@doabbreviationsdef</code>	20
<code>\@glsxtr@base@acrcmd</code>	88–94, 129, 130	<code>\@glsxtr@doaccsupp</code>	24, 27
<code>\@glsxtr@bibgls@removespaces</code>	376	<code>\@glsxtr@docdefsetting</code>	17, 62
<code>\@glsxtr@bookindex@atendgroup</code>		<code>\@glsxtr@docdefval</code>	17, 62, 64
.....	456, 457, 459	<code>\@glsxtr@doccounterrecord</code>	13
<code>\@glsxtr@bookindex@atsubendgroup</code> ..	458	<code>\@glsxtr@doglossary</code>	157, 158
<code>\@glsxtr@bookindex@atsubsubendgroup</code>	458	<code>\@glsxtr@doiflabelinlist</code>	159
<code>\@glsxtr@bookindex@between</code> .	456, 457, 459	<code>\@glsxtr@doloadprefix</code>	24, 27, 30
<code>\@glsxtr@bookindex@sep</code>	456–458	<code>\@glsxtr@doloctag</code>	70, 71
<code>\@glsxtr@bookindex@subatendgroup</code> ..		<code>\@glsxtr@dorecord</code>	8, 10
.....	456, 457, 459	<code>\@glsxtr@dorecordnodefer</code>	8, 10
<code>\@glsxtr@bookindex@subbetween</code> .	456–458	<code>\@glsxtr@dosee@alsoindex@glossary</code> ..	16
<code>\@glsxtr@bookindex@subsep</code>	456–458	<code>\@glsxtr@doseeglossary</code>	16, 29
<code>\@glsxtr@bookindex@subsubatendgroup</code>		<code>\@glsxtr@dostylewarn</code>	237
.....	457, 459	<code>\@glsxtr@enabletagging</code>	209
<code>\@glsxtr@bookindex@subsubbetween</code> ..		<code>\@glsxtr@end@</code>	65
.....	456–458	<code>\@glsxtr@endescspch</code>	204–207
<code>\@glsxtr@bookindex@groupskip</code> ...	457, 459	<code>\@glsxtr@entrycount@org@localreset</code>	116
<code>\@glsxtr@cat</code>	115, 126, 165, 209	<code>\@glsxtr@entrycount@org@localunset</code>	116
<code>\@glsxtr@check@bibgls@nameref</code>	151	<code>\@glsxtr@entrycount@org@reset</code>	116
<code>\@glsxtr@checkgroup</code>	158	<code>\@glsxtr@entrycount@org@unset</code>	115
<code>\@glsxtr@counterrecordhook</code>	11–13	<code>\@glsxtr@entryunitcount@org@localreset</code>	
<code>\@glsxtr@csname</code>	121, 122, 124	124

\@glsxtr@entryunitcount@org@localunset	\@glsxtr@nopostpunc	136
..... 124	\@glsxtr@nopostpunc@postdesc	137
\@glsxtr@entryunitcount@org@reset	\@glsxtr@notfoundinlist	215
\@glsxtr@entryunitcount@org@unset	\@glsxtr@op@recordcounter	15, 16
..... 123, 124	\@glsxtr@optlist	67
\@glsxtr@equationsfalse	\@glsxtr@org@starttoc	347
\@glsxtr@err@undefaction	\@glsxtr@org@GLS@	74
\@glsxtr@field@linkdefs	\@glsxtr@org@GLSpl@	74
\@glsxtr@floatsfalse	\@glsxtr@org@Gls@	73
\@glsxtr@format@overridefalse	\@glsxtr@org@Glspl@	73, 74
\@glsxtr@format@overridetrue	\@glsxtr@org@Glsxtrtitlefirst	348, 349
\@glsxtr@foundinlist	\@glsxtr@org@Glsxtrtitlefirstplural	
\@glsxtr@full 348, 349	
\@glsxtr@fullpl	\@glsxtr@org@Glsxtrtitlefull	348, 349
\@glsxtr@get@prefixedlabel	\@glsxtr@org@Glsxtrtitlefullpl	348, 349
.... 381–383	\@glsxtr@org@Glsxtrtitlelong	348, 349
\@glsxtr@gettype	\@glsxtr@org@Glsxtrtitlelongpl	348, 349
\@glsxtr@glossdescfont	\@glsxtr@org@Glsxtrtitlename	348, 349
..... 195, 196	\@glsxtr@org@Glsxtrtitleplural	348, 349
\@glsxtr@glossnamefont	\@glsxtr@org@Glsxtrtitleshort	348, 349
..... 197–200, 202	\@glsxtr@org@Glsxtrtitleshortpl	348, 349
\@glsxtr@glosssymbolfont	\@glsxtr@org@Glsxtrtitletext	348, 349
..... 208	\@glsxtr@org@MakeUppercase	348, 349
\@glsxtr@gobbleto@endescspch	\@glsxtr@org@addtoacronymlists	127, 130
..... 207	\@glsxtr@org@checkfirsthyper	96, 130
\@glsxtr@groupheading	\@glsxtr@org@currentfieldvalue	44
..... 158, 160	\@glsxtr@org@delimN	70, 71
\@glsxtr@idx@displaynumberlist	\@glsxtr@org@delimR	70, 71
.... 133	\@glsxtr@org@doseeglossary	29, 132
\@glsxtr@idx@entrynumberlist	\@glsxtr@org@gloautosee	29
..... 134	\@glsxtr@org@glolinkprefix	77, 79
\@glsxtr@if@record@only	\@glsxtr@org@gls@	73
..... 131, 153	\@glsxtr@org@glsdohypertarget	138
\@glsxtr@ifcsstart	\@glsxtr@org@glsignore	70, 71
..... 65	\@glsxtr@org@glspl@	73
\@glsxtr@ifnum@mmode	\@glsxtr@org@glsxtrtitlefirst	348, 349
..... 76	\@glsxtr@org@glsxtrtitlefirstplural	
\@glsxtr@ifpunctoken 348, 349	
..... 215	\@glsxtr@org@glsxtrtitlefull	348, 349
\@glsxtr@ifunitcounter	\@glsxtr@org@glsxtrtitlefullpl	348, 349
..... 121	\@glsxtr@org@glsxtrtitlelong	348, 349
\@glsxtr@insert@dots	\@glsxtr@org@glsxtrtitlelongpl	348, 349
..... 217	\@glsxtr@org@glsxtrtitlename	348, 349
\@glsxtr@insert@dots@next	\@glsxtr@org@glsxtrtitleorpdforheading	
..... 217 348, 349	
\@glsxtr@insertdots	\@glsxtr@org@glsxtrtitleplural	348, 349
..... 182, 219	\@glsxtr@org@glsxtrtitleshort	348, 349
\@glsxtr@label	\@glsxtr@org@glsxtrtitleshortpl	348, 349
..... 37, 61, 62, 171, 194	\@glsxtr@org@glsxtrtitletext	348, 349
\@glsxtr@label@prefixes		
..... 380, 381		
\@glsxtr@loadstyles		
..... 416, 417		
\@glsxtr@local@textformat		
..... 77–79		
\@glsxtr@locale		
..... 44, 45		
\@glsxtr@longnewglossaryentry		
..... 48		
\@glsxtr@mark@wordseps		
..... 217		
\@glsxtr@mark@wordseps@next		
..... 217		
\@glsxtr@markwordseps		
..... 219, 220		
\@glsxtr@mixed@assign@sortkey		
..... 134		
\@glsxtr@newglslike		
..... 162		
\@glsxtr@noidx@displaynumberlist		
.. 133		
\@glsxtr@noidx@do		
..... 160		
\@glsxtr@noidx@entrynumberlist		
.... 134		
\@glsxtr@noidx@numberlistloop		
.... 134		
\@glsxtr@nomissingglstextnr		
..... 24		
\@glsxtr@nomissingglstextval		
..... 24		
\@glsxtr@noop@recordcounter		
..... 13, 16		

<code>\@glxtr@org@makeglossaries</code>	131	<code>\@glxtr@redefstyles</code>	24, 25, 370
<code>\@glxtr@org@markboth</code>	346, 347	<code>\@glxtr@reg@glosslist</code>	132–135, 138
<code>\@glxtr@org@markright</code>	346, 347	<code>\@glxtr@restore@postpunc</code>	137
<code>\@glxtr@org@newacronymstyle</code> ..	128, 130	<code>\@glxtr@rglstrigger@record</code> ...	167–169
<code>\@glxtr@org@postdescription</code> ..	137, 211	<code>\@glxtr@s@longnewglossaryentry</code>	48
<code>\@glxtr@org@see@noindex</code>	150, 151	<code>\@glxtr@savepreloctag</code>	70, 71
<code>\@glxtr@org@setacronymlists</code>	130	<code>\@glxtr@setentrycountunsetattr</code> ...	114
<code>\@glxtr@org@setacronymstyle</code> ..	128, 130	<code>\@glxtr@setentryunitcountunsetattr</code>	126
<code>\@glxtr@org@theHvalue</code>	8–10	<code>\@glxtr@setupshortcuts</code>	22, 23, 29, 30
<code>\@glxtr@org@unset@buffer</code>	112, 113	<code>\@glxtr@shortcutsnr</code>	22
<code>\@glxtr@org@prefix</code>	11	<code>\@glxtr@shortcutsval</code>	22, 153
<code>\@glxtr@org@printglossary</code>	67, 137	<code>\@glxtr@swaptwo</code>	216
<code>\@glxtr@org@warndep</code>	218	<code>\@glxtr@tag</code>	209
<code>\@glxtr@p@acrlong@</code>	103	<code>\@glxtr@taggingcs</code>	209
<code>\@glxtr@p@acrlongpl@</code>	103	<code>\@glxtr@textformat</code>	78, 79
<code>\@glxtr@p@acrshort@</code>	103	<code>\@glxtr@theHentrycounter</code>	11
<code>\@glxtr@p@acrshortpl@</code>	103	<code>\@glxtr@theHvalue</code> ...	8–10, 75, 77–80, 166
<code>\@glxtr@p@long@</code>	103	<code>\@glxtr@theentrycounter</code>	11
<code>\@glxtr@p@longpl@</code>	103	<code>\@glxtr@thevalue</code>	8–10, 75, 77–80, 166
<code>\@glxtr@p@plural@</code>	103	<code>\@glxtr@thisloctag</code>	71
<code>\@glxtr@p@short@</code>	103	<code>\@glxtr@titlelabel</code>	140, 141, 160
<code>\@glxtr@p@shortpl@</code>	103	<code>\@glxtr@tmp</code>	25, 76, 143, 144
<code>\@glxtr@p@text@</code>	103	<code>\@glxtr@type</code>	194
<code>\@glxtr@pagetag</code>	70, 71	<code>\@glxtr@unitcountlist</code>	121
<code>\@glxtr@pagetag</code>	70, 71	<code>\@glxtr@unset</code>	112, 113
<code>\@glxtr@prefix</code>	381	<code>\@glxtr@unset@buffer</code>	112, 113
<code>\@glxtr@prevunitcount</code>	123	<code>\@glxtr@unsrt@getgroupptitle</code>	157
<code>\@glxtr@printglossnr</code>	137	<code>\@glxtr@use@equation@counter</code> .	9, 76, 78
<code>\@glxtr@printglossopts</code>	67, 134, 137	<code>\@glxtr@usesee</code>	54
<code>\@glxtr@printglossval</code>	137	<code>\@glxtr@warn@onexistsordo</code>	7, 15, 16
<code>\@glxtr@printunsrtglossaryskipentry</code>	157, 158	<code>\@glxtr@warn@undefaction</code>	7, 15, 16
<code>\@glxtr@provide@addstoragekey</code>	33	<code>\@glxtr@wrglossary@locationhyperlink</code>	26
<code>\@glxtr@provide@storagekey</code>	32	<code>\@glxtr@wrglossnr</code>	75
<code>\@glxtr@providenewgls</code>	162	<code>\@glxtr@wrglossval</code>	75
<code>\@glxtr@record</code>		<code>\@glxtr@buffer@nodup@unset</code>	112
.....	15, 16, 73, 74, 80, 223, 225–233	<code>\@glxtr@buffer@unset</code>	112
<code>\@glxtr@record@noglossarywarning</code> ..	15	<code>\@glxtr@dialecthook</code>	370
<code>\@glxtr@record@only@setup</code>	16	<code>\@glxtr@docdeffalse</code>	64
<code>\@glxtr@record@setting</code>	8, 10–12, 15, 31, 58, 63, 64, 100, 131, 151–153, 162	<code>\@glxtr@entryfmt</code>	35
<code>\@glxtr@record@setting@alsoindex</code> .		<code>\@glxtr@fmt</code>	34
.....	8, 10, 58, 131	<code>\@glxtr@glossentry</code>	154
<code>\@glxtr@record@setting@nameref</code> ...		<code>\@glxtr@glossentryother</code>	155, 156
.....	11, 12, 15, 31, 151, 152	<code>\@glxtr@hypernameprefix</code>	138, 159
<code>\@glxtr@record@setting@off</code> .	63, 100, 162	<code>\@glxtr@ifhasfield</code>	37, 38, 42, 43
<code>\@glxtr@record@setting@only</code>	15, 31	<code>\@glxtr@ifhyphenstart</code>	328
<code>\@glxtr@recordsee</code>	15, 29, 58	<code>\@glxtr@indexaliased</code>	97
<code>\@glxtr@redef@forglsentries</code>	7, 30	<code>\@glxtr@indexcrossrefsfalse</code>	17
		<code>\@glxtr@indexcrossrefstrue</code>	18

<code>\@glxstrinmark</code>	346, 347	<code>\@org@newglossaryentryprehook</code>	48
<code>\@glxstrlong</code>	103, 228	<code>\@print@unsrt@glossary</code>	156, 157
<code>\@glxstrlongpl</code>	103, 232	<code>\@printgloss@setsort</code>	134, 136
<code>\@glxstrnewgls</code>	163, 164	<code>\@printglossary</code>	67, 156, 157
<code>\@glxstrnewgls@inner</code>	162, 163	<code>\@printunsrt@glossary@handler</code>	158
<code>\@glxstrnewgls@innercsname</code>	163	<code>\@printunsrtglossary</code>	156
<code>\@glxstrnotinmark</code>	346, 347	<code>\@rGLS</code>	168
<code>\@glxstrp</code>	108, 109	<code>\@rGLS@</code>	168
<code>\@glxstrp@opt</code>	106	<code>\@rGLSpl</code>	168
<code>\@glxstrpl</code>	66, 67	<code>\@rGLSpl@</code>	168
<code>\@glxstrpostloctag</code>	69, 70, 72	<code>\@rGls</code>	167
<code>\@glxstrpreloctag</code>	69, 70, 72	<code>\@rGls@</code>	167
<code>\@glxstrsetaliasnoindex</code>	97, 98	<code>\@rGlspl</code>	168
<code>\@glxstrshort</code>	103, 226	<code>\@rGlspl@</code>	168
<code>\@glxstrshortpl</code>	103, 230	<code>\@rc@ifdefinable</code>	372
<code>\@glxstrundeftag</code>	6, 31	<code>\@rgls</code>	166
<code>\@glxstrwrglossmark</code>	27	<code>\@rgls@</code>	166
<code>\@gobble</code>	7, 16, 18, 81, 129, 132, 158, 159, 217, 456–459	<code>\@rglspl</code>	167
<code>\@gobbletwo</code>	130, 218	<code>\@rglspl@</code>	167
<code>\@gtempa</code>	372	<code>\@sGlsXtrEnableOnTheFly</code>	64
<code>\@ifdefinable</code>	372	<code>\@secondofthree</code>	81– 83, 89–92, 95, 224, 225, 227, 229, 231, 233
<code>\@ifnextchar</code>	99	<code>\@secondoftwo</code>	74, 81, 84–93, 96, 102, 130, 137, 201, 215, 223, 224, 227–233, 249, 251, 274, 276, 291, 293, 315, 317, 347, 349
<code>\@ifpackageloaded</code>	5, 20, 153, 171, 195, 196, 199, 201, 203, 370, 384, 415	<code>\@sglsxtr@provide@storagekey</code>	32
<code>\@ifstar</code>	32, 34, 37–39, 42, 43, 48–50, 64, 99, 112, 156, 209	<code>\@star@or@long</code>	372
<code>\@ifundefined</code>	369, 372	<code>\@starttoc</code>	347
<code>\@ignored@glossaries</code>	49–51	<code>\@thirdofthree</code>	81–84, 89, 90, 92, 93, 95, 224, 226, 228, 230, 232, 233, 348
<code>\@input</code>	150	<code>\@thirdoftwo</code>	84–88
<code>\@input@</code>	145	<code>\@this@key</code>	201
<code>\@istfilename</code>	132	<code>\@tracklang@lang</code>	45
<code>\@makeglossary</code>	132	<code>\@warn@nomakeglossaries</code>	146
<code>\@mfu@domakefirstuc</code>	209, 210	<code>\@xdy@main@language</code>	145
<code>\@mfu@nocaplist</code>	210	<code>\@xdycrossrefhook</code>	57, 58
<code>\@ne</code>	117, 125, 163	<code>\@xdy@language</code>	145, 146
<code>\@newglossaryentry@defcounters</code>	115, 123	<code>\@xdy@locationclassorder</code>	58
<code>\@newglossaryentryposthook</code>	14, 33, 59, 94	<code>[package</code>	374
<code>\@newglossaryentryprehook</code>	14, 33, 48, 59, 94	<code>\@</code>	144, 376
<code>\@nil</code>	144, 160, 376		
<code>\@nnil</code>	204, 205, 207, 215, 217		
<code>\@no@glxstrindexaliased</code>	97, 98		
<code>\@no@makeglossaries</code>	149		
<code>\@nocounterr</code>	171		
<code>\@nopostdesc</code>	136		
<code>\@onelevel@sanitize</code>	11, 13, 58, 67, 141, 160		
<code>\@onlypreamble</code>	68, 70, 125, 151, 154, 171, 203, 205, 206, 209		
<code>\@org@glossaryentrynumbers</code>	135, 136		

A

<code>\AA</code>	399
<code>\aa</code>	399
<code>\AB</code>	21
<code>\Ab</code>	21
<code>\ab</code>	20

abbreviation styles:	\acp	21
footnote	\ACRfull	93
long-hyphen-postshort-hyphen	\Acrfull	93
long-hyphen-short-hyphen	\acrfull	93
long-postshort-user	\ACRfullfmt	93, 128
long-short	\Acrfullfmt	93, 128
long-short-user	\acrfullfmt	93, 128
nolong-short	\ACRfullpl	94
short	\Acrfullpl	93
short-em-footnote	\acrfullpl	93
short-em-postfootnote	\ACRfullplfmt	94, 128
short-footnote	\Acrfullplfmt	94, 128
short-hyphen-long-hyphen	\acrfullplfmt	93, 128
short-hyphen-postlong-hyphen	\ACRlong	91
short-long-user	\Acrlong	91
short-nolong	\acrlong	91
short-nolong-desc	\ACRlongpl	92
short-postfootnote	\Acrlongpl	92
short-postlong-user	\acrlongpl	92
short-sc-footnote	\acronymentry	127
short-sm-postfootnote	\acronymfont	89–93, 105, 106, 129, 130
\abbreviationsname	\acronymname	20
\abbrvpluralsuffix	\acronymsort	127
	\acronymtype	20, 127, 129, 370
	\acrpluralsuffix	127, 152
	\ACRshort	89
	\Acrshort	89
	\acrshort	88
	\ACRshortpl	90
\ABP	\Acrshortpl	90
\Abp	\acrshortpl	90
\abp	\ACS	21
\AC	\Acs	21
\Ac	\acs	21
\ac	\ACSP	21
\ACF	\Acsp	21
\Acf	\acsp	21
\acf	\actualchar	206
\ACFP	\addtolength	446
\Acfp	\advance	117, 125, 151, 163
\acfp	\AF	21
\ACL	\Af	21
\Acl	\af	20
\acl	\AFP	21
\ACLP	\Afp	21
\Aclp	\afp	21
\aclp	\AL	21
\ACP	\Al	21
\Acp	\al	20

<code>\ALP</code>	21	<code>glossnamefont</code>	197, 199
<code>\Alp</code>	21	<code>headuc</code>	349
<code>\alp</code>	20	<code>indexname</code>	204
<code>\alsoname</code>	57	<code>indexonlyfirst</code>	98
<code>amsmath package</code>	12, 26	<code>insertdots</code>	219
<code>\AnyTrackedLanguages</code>	370, 415	<code>linkcount</code>	170
<code>\appto</code>	14, 25, 33, 53, 54, 58–60, 94, 98, 99, 115, 123, 158, 159, 203, 214, 217, 219, 380, 416	<code>linkcountmaster</code>	170
<code>\apptoglossarypreamble</code>	377–379	<code>markshortwords</code>	219
<code>\arabic</code>	26, 459	<code>markwords</code>	218, 220, 328, 329, 338
<code>\AS</code>	21	<code>nameshortaccess</code>	183, 184
<code>\As</code>	21	<code>nohyper</code>	96
<code>\as</code>	20	<code>nohyperfirst</code>	82–84
<code>\ASP</code>	21	<code>noshortplural</code>	219
<code>\Asp</code>	21	<code>regular</code>	72, 120, 240–244, 246, 248, 250, 252–256, 258, 259, 261–263, 265, 266, 269–271, 273–275, 278–281, 283, 285, 287, 288, 290, 292, 295–297, 299, 301–303, 306, 308, 310–315, 317, 319, 325–327, 329–331, 333, 338, 339, 344, 345
<code>\asp</code>	20	<code>textformat</code>	78
<code>\AtBeginDocument</code>	27, 31, 68, 69, 153, 381	<code>textshortaccess</code>	183
<code>\AtEndDocument</code>	61, 116, 124, 145, 146		
B			
<code>babel package</code>	203, 205, 214	<code>\cdot</code>	27
<code>\begin</code>	142, 147, 148, 157, 418, 420–427, 448, 450, 451, 453, 456, 465–479, 481, 482, 488, 489	<code>\centering</code>	455
<code>\begingroup</code>	8, 9, 34, 36, 77, 80, 97, 155–157, 170, 181, 208, 372, 375, 376, 381	<code>\cGLS</code>	21, 114, 126
<code>\bgroup</code>	48, 136	<code>\cGls</code>	21, 114, 126
<code>bib2gls</code> .	26, 34, 47, 100, 151, 160, 162, 165, 166, 369–372, 374, 377, 379, 381, 383, 490	<code>\cgl</code>	20, 21, 114, 126
<code>\bibglsdelimN</code>	371	<code>\cGLSformat</code>	118
<code>\bibglshrefchar</code>	151	<code>\cGlsformat</code>	118
<code>\bibglslastDelimN</code>	371	<code>\cglformat</code>	117, 119
<code>\bottomrule</code>	463, 467– 469, 471, 472, 474, 476, 477, 479, 480, 482	<code>\cGLSpl</code>	21, 114, 126
C			
<code>\c@wrglossary</code>	26	<code>\cGlspl</code>	21, 114, 126
<code>\catcode</code>	62, 150	<code>\cglspl</code>	20, 21, 114, 126
category attributes:		<code>\cGLSplformat</code>	119
<code>accessinsertdots</code>	181	<code>\cGlsplformat</code>	118
<code>apospplural</code>	219	<code>\cglsplformat</code>	118, 120
<code>discardperiod</code>	213	<code>\char</code>	141
<code>entrycount</code>	111, 114–116, 126	<code>\columnwidth</code>	69
<code>externallocation</code>	374	<code>\count@</code>	116, 117, 125, 126
<code>firstshortaccess</code>	183, 245, 248	<code>\csappto</code>	46
<code>firstuc</code>	199	<code>\csdef</code>	33, 41, 46, 47, 94, 95, 116, 121, 122, 124, 184, 189, 201, 211, 212, 237–239, 248, 250, 274, 275, 290, 292, 315, 317, 320, 322, 323, 325, 335, 337, 340, 342, 417, 436
<code>glossdesc</code>	195	<code>\cseappto</code>	52
<code>glossdescfont</code>	195	<code>\csedef</code>	122, 487
<code>glossname</code>	196	<code>\csgdef</code>	42, 49–51, 63, 70, 116, 122, 124, 125, 129, 436, 437, 459
		<code>\cslet</code>	41, 48, 136

<code>\csletcs</code>	41, 239	<code>\descriptionname</code>	463, 466, 468, 469, 471, 472, 474, 475, 477, 479, 480, 482
<code>\csname</code>	6, 33, 50, 58, 63, 68, 72, 74, 77, 80, 89–95, 97, 106, 122, 132, 133, 142, 145, 146, 149, 150, 157, 163, 164, 166, 170, 171, 194, 218, 223–234, 239, 240, 445	<code>\detokenize</code>	65
<code>\cspreto</code>	47	<code>\dimen@</code>	131, 436–444, 464
<code>\csuse</code>	9, 12, 35, 41, 47, 50, 60, 70, 94, 95, 108–110, 121– 123, 125, 135, 140, 142, 143, 154, 155, 159–161, 189, 200, 211, 212, 237, 239, 375, 377, 380, 436, 437, 439, 440, 463, 487	<code>\dimen@i</code>	439–441
<code>\csxdef</code>	53, 59, 122, 125	<code>\dimen@ii</code>	436, 437, 439–441, 464
<code>\currentglossary</code> ...	136, 155, 156, 456, 459	<code>\dimexpr</code>	68, 69, 435, 464, 465, 486
<code>\CurrentOption</code>	27, 416, 417	<code>\disable@keys</code>	20, 30, 31, 64, 150
<code>\CurrentTrackedLanguage</code>	415	<code>\do</code>	6, 25, 37, 61, 80, 114, 115, 126, 129, 130, 132, 135, 142, 157, 165, 171, 194, 201, 209, 381
<code>\CurrentTrackedLanguageTag</code>	152	<code>\do@glsl@link@checkfirsthyper</code>	34, 73, 74, 78, 89–93, 223–233
<code>\CurrentTrackedScript</code>	415	<code>\do@glsl:disablehyperinlist</code>	78, 97
<code>\CurrentTrackedTag</code>	370, 415	<code>doc package</code>	206
<code>\CustomAbbreviationFields</code>	220, 240, 242–245, 247, 248, 250, 251, 253, 256, 259–265, 267, 271, 273, 275, 277–282, 284, 288–290, 292, 293, 295, 297–301, 303, 305, 307, 310, 312, 314, 315, 317, 319, 320, 322, 324–327, 329– 331, 333, 335, 336, 338–340, 342, 343, 345	<code>\dolistcsloop</code>	36
D		<code>\DTLifinlist</code>	37, 38, 77, 129, 132–134, 138, 381
<code>datatool-base package</code>	12	<code>\DTLifint</code>	141
<code>\DeclareAcronymList</code>	127	E	
<code>\DeclareOption</code>	5, 416	<code>\eappto</code>	13, 25, 49–51, 129, 158, 160, 182, 183, 204, 416
<code>\DeclareOptionX</code>	5, 27	<code>\edef</code>	6, 8, 9, 11, 35, 44, 49–52, 57, 59, 60, 62, 63, 77, 78, 80, 96, 97, 100, 102, 121, 122, 124, 129, 132, 133, 138, 141, 143–146, 150, 155, 156, 166, 170, 182, 195–201, 204, 205, 207, 208, 216, 237, 376, 381, 415, 439, 440, 456–458, 465–479, 481, 482, 488, 489
<code>\def</code>	9, 11, 13–17, 28, 31, 34, 36, 39, 48, 50, 54, 58, 59, 61, 65–67, 69, 72–94, 100, 102–106, 112, 117–120, 127, 133, 135, 137–139, 141–145, 157, 160, 163, 166–169, 179– 182, 204–207, 209, 210, 214–219, 223– 233, 237, 328, 331, 334, 337, 340, 374, 376, 380, 381, 415, 429, 445–447, 451– 453, 456, 457, 463, 464, 484, 485, 488, 489	<code>\eglssetwidest</code>	438–444
<code>\defglentryfmt</code>	49–51	<code>\egroup</code>	48, 136
<code>\define@boolkey</code>	17, 18, 75, 96	<code>\else</code>	8–13, 15, 17–20, 22–24, 28, 29, 31, 34, 39, 40, 46, 62, 64, 65, 70, 72, 74, 79, 97, 98, 117, 129, 131, 132, 134, 136, 137, 141, 143, 144, 147, 149, 151, 165, 166, 203–205, 207, 215, 217, 220, 227–234, 236, 241, 243, 244, 246, 247, 249, 250, 252–258, 261, 263–272, 274, 275, 277–289, 291, 292, 294, 296, 298, 300, 302–314, 316, 319–321, 323, 324, 327–329, 331, 334, 336, 337, 340, 342, 344, 345, 376–379, 418, 421–428, 430, 432, 434, 446, 453, 455–458, 466–483, 487
<code>\define@choicekey</code>	7, 15, 17, 19, 22, 24, 27, 75, 137	<code>\emph</code>	180, 293
<code>\define@key</code>	14, 19, 24, 25, 33, 59, 75, 76, 79, 94, 138, 179, 216	<code>\empty</code>	143, 144, 180, 376, 380, 381
<code>\DefineAcronymSynonyms</code>	22, 23	<code>\encapchar</code>	207
<code>\delimN</code>	70, 71	<code>\end</code>	142, 147, 148, 158, 418, 420–427, 449–451, 453, 457, 466–473, 475, 476, 478–483, 489
<code>\delimR</code>	70, 71	<code>\end@getprefix</code>	31

<code>\GlossariesWarningNoLine</code>	133, 146	<code>\glossaryname</code>	135
<code>glossary</code> styles:		<code>\glossarypostamble</code>	142, 158, 159
<code>altlist</code>	419	<code>\glossarypreamble</code>	142, 157
<code>altlistgroup</code>	419	<code>\glossarysection</code> ...	142, 149, 150, 157, 159
<code>altlisthypergroup</code>	420	<code>\glossarytitle</code> 50, 135, 136, 142, 149, 150, 157	
<code>alttree</code>	377, 378, 435, 436, 445, 463	<code>\glossarytoctitle</code>	
<code>alttreegroup</code>	446	50, 135, 138, 142, 149, 150, 157
<code>alttreehypergroup</code>	447	<code>\glossentry</code> ..	136, 161, 417–427, 429, 432,
<code>index</code>	429	434, 445, 457, 466, 467, 469, 470, 472,	
<code>indexgroup</code>	430	473, 475, 476, 478, 480, 481, 483, 484, 488	
<code>indexhypergroup</code>	430	<code>\Glossentrydesc</code>	486
<code>inline</code>	428	<code>\glossentrydesc</code>	
<code>list</code>	418, 419	417, 418, 420–428, 431–435, 461, 487
<code>listdotted</code>	417	<code>\Glossentryname</code>	485
<code>listdottedstyle</code>	418	<code>\glossentryname</code>	155,
<code>listgroup</code>	419	417–427, 430, 432, 434, 446, 454, 461, 487	
<code>listhypergroup</code>	419	<code>\glossentrynameother</code>	156
<code>longragged-booktabs</code>	465	<code>\glossentrysymbol</code>	422,
<code>mcolalttree</code>	451	424, 426, 428, 431, 433, 435, 461, 485, 487	
<code>mcolalttreegroup</code>	452	<code>\glossxtrsetopts</code>	211
<code>mcolalttreehypergroup</code>	452	<code>\GLS</code>	114, 126, 165
<code>mcolalttreespannav</code>	453	<code>\Gls</code>	66, 114, 126, 165
<code>mcolindexgroup</code>	448	<code>\gls</code>	46, 66, 67, 114, 126, 133, 147, 165
<code>mcolindexhypergroup</code>	448	<code>\gls@assign@desc</code>	48
<code>mcolindexspannav</code>	448	<code>\gls@assign@field</code>	14, 33, 94
<code>mcoltreegroup</code>	449	<code>\gls@checkseeallowed</code>	62, 64, 132
<code>mcoltreehypergroup</code>	449	<code>\gls@codepage</code>	146
<code>mcoltreenonamegroup</code>	450	<code>\gls@defdocnewglossaryentry</code> ..	62, 115, 123
<code>mcoltreenonamehypergroup</code>	450	<code>\gls@defglossaryentry</code>	48, 63, 66, 67
<code>mcoltreenonamespannav</code>	451	<code>\gls@dotocitle</code>	135, 136
<code>mcoltreespannav</code>	450	<code>\gls@glossary</code>	58
<code>name-desc</code>	468	<code>\gls@grplabel</code>	100
<code>sublistdotted</code>	418	<code>\gls@ifnotmeasuring</code>	12, 114
<code>tree</code>	431	<code>\gls@level</code>	161
<code>treegroup</code>	433	<code>\gls@noidxglossary</code>	133
<code>treehypergroup</code>	433	<code>\gls@org@glossaryentryfield</code>	136
<code>treenoname</code>	433	<code>\gls@org@glossarysubentryfield</code>	136
<code>treenonamegroup</code>	434	<code>\gls@orgTrackLangRequireDialectPrefix</code>	
<code>treenonamehypergroup</code>	435	415
<code>glossary-bookindex</code> package	417	<code>\gls@save@numberlist</code>	69, 70, 72
<code>glossary-hypernav</code> package	100	<code>\gls@set@xr@key</code>	58, 59
<code>glossary-long</code> package	422	<code>\gls@tmplen</code>	438–444, 446, 464, 465
<code>glossary-longbooktabs</code> package	422	<code>\gls@type</code>	133
<code>glossary-longextra</code> package	377–379	<code>\glsabbrvdefaultfont</code> ...	222, 241, 243,
<code>glossary-tree</code> package	378, 379, 429, 486	246, 249, 252, 254, 257, 318, 329, 332, 343	
<code>\glossaryentrynumbers</code> ...	72, 135, 136, 161	<code>\glsabbrvemfont</code>	
<code>\glossaryheader</code> .	142, 157, 418–427, 429,	293–301, 303, 305, 307, 309, 310, 312–317	
	430, 432–435, 445, 447–450, 452, 457,	<code>\glsabbrvfont</code>	104, 105, 129, 227,
	466, 467, 469–473, 475, 476, 478–484, 488	228, 230–232, 234, 236, 240–254, 257,	

258, 260, 262, 264, 266, 268, 269, 271, 274, 277, 279, 281, 282, 284, 286, 288, 291, 294, 296, 298, 300, 301, 303, 305, 307, 309, 310, 313, 316, 319, 321, 323, 325–327, 330, 332, 334–336, 338, 341, 344	\glsabbrvhyphenfont 329, 330, 335, 337–342	\glsaccessplural 83
\glsabbrvonlyfont 343–345	\Glsaccessshort 89, 227, 236, 243, 246, 247, 249, 250, 254, 256, 263, 265, 266, 272, 274, 275, 279, 281, 283, 289, 291, 292, 298, 300, 302, 303, 313, 314, 316, 323, 324, 327, 335, 341, 342	\glsaccessshort 89, 221, 227, 228, 236, 241, 243, 246, 247, 249, 252, 254, 255, 257, 261, 263–268, 270, 272, 274, 275, 277–279, 281–285, 287–289, 291, 294, 296, 298, 300–304, 306–309, 311, 313, 316, 319–321, 323, 324, 327, 330, 332, 335, 338, 341, 342, 344
\glsabbrvscfont 260–266, 268, 269, 271, 273–275	\Glsaccessshortpl 90, 231, 236, 244, 246, 247, 249, 250, 254, 256, 263, 265, 266, 272, 275, 280, 281, 283, 289, 291, 292, 298, 300, 302, 303, 313, 314, 316, 323, 324, 327, 336, 341, 342, 345	\glsaccessshortpl 90, 221, 230, 232, 236, 241, 243, 246, 247, 249, 250, 252, 254, 255, 257, 261, 263–270, 272, 274, 275, 277–279, 281–285, 287–289, 291, 294, 296, 298, 300, 302–304, 306–311, 313, 314, 316, 319–321, 323, 324, 327, 330, 332, 335, 338, 339, 341, 342, 344, 345
\glsabbrvsmfont 276–282, 284, 286, 288, 290–292	\GLSaccesssymbol 86	\GLSaccesssymbol 85, 208
\glsabbrvuserfont 318–324, 326	\Glsaccesssymbol 85, 208, 213	\glsaccesssymbol 85, 208, 213
\GLSaccessdesc 85	\GLSaccesssymbolplural 86	\GLSaccesssymbolplural 86
\Glsaccessdesc 84, 195, 207	\Glsaccesssymbolplural 86	\glsaccesssymbolplural 86
\glsaccessdesc 84, 195, 213	\GLSaccessstext 81	\GLSaccessstext 81
\GLSaccessdescplural 85	\Glsaccessstext 81	\glsaccessstext 81
\glsaccessdescplural 85	\glsacrshortcutstrue 22, 23	\glsacrshortcutstrue 22, 23
\GLSaccessfirst 82	\glsacspacemax 131	\glsacspacemax 131
\Glsaccessfirst 82	\glsadd 34, 62, 77, 80, 147	\glsadd 34, 62, 77, 80, 147
\glsaccessfirst 82	\glsadd options	\glsadd options
\GLSaccessfirstplural 84	theHvalue 10	theHvalue 10
\Glsaccessfirstplural 83	thevalue 9, 10	thevalue 9, 10
\glsaccessfirstplural 83	\glsaddpostsetkeys 10, 80	\glsaddpostsetkeys 10, 80
\Glsaccesslong 91, 221, 229, 241, 252, 257, 261, 267, 268, 270, 278, 284–287, 294, 296, 304–309, 311, 320, 321, 330, 332, 333, 336, 338, 344, 345	\glsaddpresetkeys 10, 80	\glsaddpresetkeys 10, 80
\glsaccesslong 91, 92, 221, 229, 230, 241, 243, 244, 246, 247, 250, 252, 254, 255, 257, 258, 261, 263–270, 272, 275, 277, 279–281, 283–289, 291, 292, 294, 296, 298, 300, 302–314, 316, 319, 321, 324, 327, 330, 332, 336, 338, 344	\glsaddstoragekey 60, 189, 373, 374	\glsaddstoragekey 60, 189, 373, 374
\Glsaccesslongpl 92, 221, 233, 241, 252, 257, 261, 267– 270, 278, 284–287, 294, 296, 304–311, 320, 321, 330, 332, 333, 336, 339, 344, 345	\glsbackslash 65	\glsbackslash 65
\glsaccesslongpl 92, 93, 221, 232, 233, 241, 243, 244, 246, 247, 250, 252, 254–258, 261, 263– 272, 275, 277, 279–281, 283–289, 292, 294, 296, 298, 300, 302–314, 316, 319, 321, 324, 327, 330, 332, 333, 336, 338, 344	\glsapscase 74, 81–93, 95, 223–235	\glsapscase 74, 81–93, 95, 223–235
\GLSaccessname 84	\glsacategory 72, 81, 96, 104, 105, 190–192, 195–201, 207, 208, 211, 212, 223–225, 227, 228, 230–232	\glsacategory 72, 81, 96, 104, 105, 190–192, 195–201, 207, 208, 211, 212, 223–225, 227, 228, 230–232
\Glsaccessname 84		
\glsaccessname 84		
\GLSaccessplural 83		
\Glsaccessplural 83		

<code>\glscategorylabel</code>	96,	<code>\Glsentryfirstplural</code> ...	120, 169, 174, 186
180, 183, 216, 218–220, 237, 240, 242–		<code>\glstryfirstplural</code>	120, 169, 174, 186, 365
245, 247, 248, 250, 251, 253, 259–265,		<code>\glstryfmt</code>	49–51, 129
271, 273–275, 277–280, 282, 284, 288–		<code>\Glsentryfull</code>	128
290, 292, 293, 295–301, 303, 305, 306,		<code>\glstryfull</code>	128
308, 312, 314, 315, 317, 319, 320, 322–		<code>\Glsentryfullpl</code>	128
327, 329, 330, 333, 335–340, 342, 343, 345		<code>\glstryfullpl</code>	128
<code>\glsclosebrace</code>	58, 148, 149	<code>\glstryitem</code>	155, 156, 417–
<code>\glscounter</code>	9, 18	427, 429, 432, 434, 446, 457, 461, 485, 487	
<code>\glscurrententrylabel</code>		<code>\Glsentrylong</code> ...	105, 106, 120, 169, 178, 188
..... 69–71, 136, 145, 155–158, 210, 211		<code>\glstrylong</code>	105, 106,
<code>\glscurrentfieldvalue</code>		120, 169, 178, 188, 249, 250, 274, 276,	
34, 35, 37–40, 42–44, 55, 56, 318, 372, 373		290, 292, 315, 317, 323, 325, 340, 365–367	
<code>\glscustomtext</code> ..	73, 74, 89–93, 223–234, 236	<code>\Glsentrylongpl</code>	105, 106, 120, 178, 188
<code>\glsglobalshortaccess</code>	182	<code>\glstrylongpl</code>	
<code>\glsglobaltype</code> 105, 106, 120, 178, 179, 188, 366, 367	
..... 6, 20, 46, 47, 135, 147, 156, 157, 159		<code>\Glsentrylongplural</code>	169
<code>\glsglobalaccessdisplay</code> ..	176, 195	<code>\glstrylongplural</code>	169
<code>\glsglobalpluralaccessdisplay</code>		<code>\Glsentryname</code>	172, 185, 197–200
..... 176, 177		<code>\glstryname</code>	154,
<code>\glsglobalwidth</code>	420–427, 463–465	155, 172, 185, 204, 361, 362, 438–444, 460	
<code>\glsgloballabel</code>	8,	<code>\glstrynumberlist</code>	134, 140, 443, 444
9, 32, 35–42, 46–48, 52–54, 56, 57, 61,		<code>\glstrypdfsymbol</code>	208
63, 65, 77, 80, 97, 101, 115, 116, 121–		<code>\Glsentryplural</code>	173, 186
125, 132, 136, 139, 140, 155, 156, 160,		<code>\glstryplural</code>	173, 186, 363
161, 164, 166, 194, 197–199, 202, 439, 440		<code>\glstryprevcount</code>	115, 117, 123
<code>\glsglobal</code>	383	<code>\glstryprevmaxcount</code>	123
<code>\glsglobalnumberlist</code>	133, 139	<code>\glstryprevtotalcount</code>	123
<code>\glsglobalhyperlink</code>	100, 377	<code>\Glsentryshort</code>	104, 105, 177, 187
<code>\glsglobalhypertarget</code>	102, 137	<code>\glstryshort</code>	104, 105,
<code>\glsglobalexists</code>	17, 28, 40,	131, 177, 187, 320, 322, 334, 360, 361, 367	
41, 48, 52, 54–58, 73, 74, 80, 88–93, 102,		<code>\Glsentryshortpl</code>	105, 178, 188
114, 132, 139, 140, 155, 156, 223–233, 371		<code>\glstryshortpl</code>	
<code>\glsglobalexistsordo</code>	34, 35, 74	104–106, 177, 178, 187, 188, 360, 361, 367	
<code>\glsglobalexistsorwarn</code>	17, 195, 196, 207, 208	<code>\Glsentrysymbol</code>	175, 186
<code>\glsglobalnoexists</code>	48	<code>\glstrysymbol</code>	175, 186, 208, 441–443
<code>\glsglobalhyperlink</code>	79, 102	<code>\Glsentrysymbolplural</code>	175, 187
<code>\glsglobalsanitizesort</code>	134	<code>\glstrysymbolplural</code>	175, 186, 187
<code>\glsglobalentrycount</code>	114, 117, 125	<code>\Glsentrytext</code>	172, 185
<code>\glsglobalentryunitcount</code>	116, 126	<code>\glstrytext</code> ...	102, 172, 173, 185, 362, 363
<code>\glstrycounter</code>	26, 144, 376	<code>\glstrytype</code>	155, 221
<code>\GlsEntryCounterLabelPrefix</code>	46	<code>\Glsentryuseri</code>	86
<code>\glstrycurrcount</code>	115, 117, 123	<code>\glstryuseri</code>	87
<code>\Glsentrydesc</code>	176, 187, 196	<code>\Glsentryuserii</code>	87
<code>\glstrydesc</code>	176, 187, 196	<code>\glstryuseriii</code>	87
<code>\Glsentrydescplural</code>	176, 187	<code>\Glsentryuseriii</code>	87
<code>\glstrydescplural</code>	176, 177, 187	<code>\Glsentryuseriv</code>	87
<code>\Glsentryfirst</code>	120, 169, 174, 186	<code>\glstryuseriv</code>	87
<code>\glstryfirst</code> ..	120, 169, 173, 174, 186, 364		

\Glsentryuserv	88	\GLSfmtname	56
\glsentryuserv	88	\Glsfmtname	55, 56
\Glsentryuservi	88	\glsfmtname	54–56
\glsentryuservi	88	\GLSfmttext	56
\glsextrapostnamehook	200	\Glsfmttext	55, 56
\glsfieldfetch	101	\glsfmttext	54–56
\glsfieldxdef	194	\glsforeachincategory	237
\glsFindWidestLevelTwo	379, 380	\glsgenentryfmt	72
\glsFindWidestTopLevelName	379	\glsgetattribute	78, 101, 117, 121–123, 145, 165, 170, 195–201, 203, 208
\glsfindwidesttoplevelname	438	\glsgetcategoryattribute	190
\GLSfirst	354, 355	\glsgetgrouptitle	419, 420
\Glsfirst	355	\glsgetwidestname	436
\glsfirst	354	\glsgroupheading	
\glsfirstabbrvdefaultfont 160, 418–427, 429, 430, 432–435, 445, 447–453, 458, 466, 467, 469, 470, 472, 473, 475, 476, 478, 480, 481, 483, 484, 488
222, 241, 243, 246, 249, 252, 254, 257, 332		\glsgroupskip	160, 418, 421–428, 430, 432, 434, 446, 458, 466, 468–472, 474, 475, 477, 478, 480, 481, 483, 485, 489
\glsfirstabbrvmfont	293–317	\glshasattribute	78, 101, 116, 117, 121, 122, 124, 125, 145, 165, 170, 195–201, 203, 208, 241–244, 246, 248, 249, 251, 253, 255, 256, 258–260, 262, 263, 271, 273, 274, 276–280, 288, 290– 292, 294–297, 299, 301, 308, 312–315, 317, 319, 320, 322, 323, 325–327, 329– 331, 333, 335, 337–339, 341, 342, 344, 345
\glsfirstabbrvfont	129, 221, 240–257, 260, 262, 264, 266, 268, 269, 271, 274, 277, 279, 281, 282, 285, 286, 288, 291, 294, 296, 298, 300, 301, 303, 305, 307, 309, 310, 313, 316, 319, 321, 323, 326, 330, 332, 335, 338, 341, 344	\glshascategoryattribute	190
\glsfirstabbrvhyphenfont		\glshex ..	386–391, 394–399, 402–404, 406–415
..... 328–330, 334, 335, 337–342		\glshyperlink	102, 373
\glsfirstabbrvonlyfont	344, 345	\glshypernavsep	142
\glsfirstabbrvscfont	260–275	\glshypernumber	145, 203, 374–376
\glsfirstabbrvsmfont	277–292	\glsifattribute 75, 76, 82, 96, 98, 108, 170, 193, 195–198, 202, 210, 213, 214, 350–359	
\glsfirstabbrvuserfont	319–327	\glsifcategory	192
\glsfirstaccessdisplay	173, 174	\glsifcategoryattribute	
\glsfirstlongdefaultfont 96, 180, 183, 191, 218–220	
241, 243, 252, 254, 257, 260–271, 277– 287, 293–295, 297–299, 301–306, 309, 310		\glsifnotregular	81
\glsfirstlongemfont		\glsifnotregularcategory	192
.... 295–297, 299–301, 307, 308, 310–312		\glsifplural . 74, 81–86, 89–93, 214, 223–235	
\glsfirstlongfont 221, 240–244, 246, 249, 252, 254–259, 261, 262, 264, 266, 268, 269, 271, 274, 277, 279, 281, 282, 285, 286, 288, 291, 294, 296, 298, 300, 301, 303, 305, 307, 309, 311, 313, 316, 319, 321, 323, 326, 330, 332, 335, 338, 341, 344		\glsifregular	72, 81, 120, 169
\glsfirstlongfootnotefont		\glsifregularcategory	192
.... 245–250, 271–276, 288–292, 312–317		\glsifusetranslator	50
\glsfirstlonghyphenfont	328–341	\glsignore	70, 71
\glsfirstlongonlyfont	343–345	\glsinlinedescformat	428
\glsfirstlonguserfont	319–327	\glsinlinesubdescformat	428
\GLSfirstplural	355, 356	\glsinsert	74, 81, 89–93, 223–236, 328, 335, 337, 340–342
\Glsfirstplural	356		
\glsfirstplural	355		
\glsfirstpluralaccessdisplay	174		

<code>\glskeylisttok</code>	127, 218, 220	<code>\glslongextraDescNameTabularHeader</code>	468, 469
<code>\glslabel</code> 8, 9, 34, 35, 52, 72, 75–79, 96, 97,	101, 102, 130, 166, 170, 212, 213, 234–	<code>\glslongextraDescSymNameHeader</code>	481	
236, 249, 250, 274, 276, 290, 292, 315,	317, 320, 322, 323, 325, 335, 337, 340–342	<code>\glslongextraDescSymNameTabularFooter</code>	480, 481
<code>\glslabeltok</code>	<code>\glslongextraDescSymNameTabularHeader</code>	480, 481
..... 127, 218, 220, 221, 240–249, 251,	253–264, 266, 268, 271, 273, 274, 276–	<code>\glslongextraGroupHeading</code> 466, 467, 469,	470, 472, 473, 475, 476, 478, 480, 481, 483	
282, 284, 288, 290–301, 303, 305, 307,	308, 310, 312–315, 317, 319, 320, 322–	<code>\glslongextraHeaderFmt</code> .	463, 464, 466,	
327, 329–331, 333, 335, 337–339, 341–345		468, 469, 471, 472, 474–477, 479, 480, 482		
<code>\glsletentryfield</code>	204	<code>\glslongextraLocationAlign</code>	467, 470, 473, 476, 479, 482, 483
<code>\glslink</code>	128, 383	<code>\glslongextraLocationDescNameHeader</code>	470	
<code>\glslink options</code>		<code>\glslongextraLocationDescNameTabularFooter</code>	469, 470
counter	10	<code>\glslongextraLocationDescNameTabularHeader</code>	469, 470
format	202	<code>\glslongextraLocationDescSymNameHeader</code>	483
hyper	346	<code>\glslongextraLocationDescSymNameTabularFooter</code>	482
hyperoutside	75, 76	<code>\glslongextraLocationDescSymNameTabularHeader</code>	482
noindex	8, 9, 96, 346	<code>\glslongextraLocationFmt</code>	467, 470, 473, 477, 480, 483
textformat	78	<code>\glslongextraLocationSymDescNameHeader</code>	480
theHvalue	78	<code>\glslongextraLocationSymDescNameTabularFooter</code>	478, 479
thevalue	78, 162	<code>\glslongextraLocationSymDescNameTabularHeader</code>	478, 479
wrgloss	8, 75	<code>\glslongextraLocSetDescWidth</code> ..	467, 470	
<code>\glslinkcheckfirsthyperhook</code>	96	<code>\glslongextraNameAlign</code>	465–476, 478, 479, 481–483
<code>\glslinkpostsetkeys</code>	10, 78, 166	<code>\glslongextraNameDescHeader</code>	466	
<code>\glslinkpresetkeys</code>	10, 78, 166	<code>\glslongextraNameDescLocationHeader</code>	467	
<code>\glslinkvar</code>	99	<code>\glslongextraNameDescLocationTabularFooter</code>	466, 467
<code>\glslistchildpostlocation</code>	418	<code>\glslongextraNameDescLocationTabularHeader</code>	466, 467
<code>\glslistchildprelocation</code>	418, 419	<code>\glslongextraNameDescSymHeader</code>	472	
<code>\glslistdesc</code>	418, 419	<code>\glslongextraNameDescSymLocationHeader</code>	473
<code>\glslistdottedwidth</code>	417	<code>\glslongextraNameDescSymLocationTabularFooter</code>	472, 473
<code>\glslistgroupheaderfmt</code>	419, 420	<code>\glslongextraNameDescSymLocationTabularHeader</code>	472, 473
<code>\glslistgroupskip</code>	418			
<code>\glslistnavigationitem</code>	419, 420			
<code>\glslistprelocation</code>	418, 419			
<code>\glslocalunset</code>	74, 166			
<code>\glslongaccessdisplay</code>	178			
<code>\glslongdefaultfont</code> .	222, 223, 241, 243,			
245, 252, 254, 257, 261, 262, 264, 266,	268–270, 277, 279, 281, 282, 284–286,			
294, 298, 301, 303, 305, 309, 318, 329, 343				
<code>\glslongemfont</code>	293, 296, 299, 300, 307, 310, 311			
<code>\glslongextraDescAlign</code>			
.....	465–476, 478, 479, 481–483			
<code>\glslongextraDescFmt</code>	462, 466, 467, 469,			
470, 472, 473, 475, 477, 478, 480, 481, 483				
<code>\glslongextraDescNameHeader</code>	469			
<code>\glslongextraDescNameTabularFooter</code>	468			

<code>\glslongextraNameDescSymTabularFooter</code>	<code>\glslongfont</code>
..... 471	. 105, 223, 229, 230, 232–234,
<code>\glslongextraNameDescSymTabularHeader</code>	241–244, 246, 247, 249, 252–254, 256,
..... 471	257, 259, 261, 262, 264, 266, 268, 269,
<code>\glslongextraNameDescTabularFooter</code>	271, 274, 277, 279, 281, 282, 285, 286,
..... 463, 466	288, 291, 294, 296, 298, 300, 301, 303,
<code>\glslongextraNameDescTabularHeader</code>	305, 307, 309, 311, 313, 316, 319, 321,
..... 463, 466	323, 326, 330, 332, 335, 338, 341, 344, 345
<code>\glslongextraNameFmt</code> ... 466, 467, 469,	<code>\glslongfootnotefont</code>
470, 472, 473, 475, 477, 478, 480, 481, 483	245, 246, 249, 271, 274, 288, 291, 313, 316
<code>\glslongextraNameSymDescHeader</code> 475	<code>\glslonghyphenfont</code>
<code>\glslongextraNameSymDescLocationHeader</code> 329–333, 335, 338, 340, 341
..... 476	<code>\glslongonlyfont</code> 343, 344
<code>\glslongextraNameSymDescLocationTabularFooter</code>	<code>\glslongpltok</code> ... 220, 240, 242–245, 248,
..... 475, 476	256, 259–263, 268, 271, 273, 277–280,
<code>\glslongextraNameSymDescLocationTabularHeader</code>	284, 288, 290, 293, 295, 297, 299, 301,
..... 475, 476	305, 307, 310, 312, 314, 319, 320, 322–
<code>\glslongextraNameSymDescTabularFooter</code>	327, 329–331, 333, 335, 337–339, 343, 345
..... 474	<code>\glslongpluralaccessdisplay</code> ... 178, 179
<code>\glslongextraNameSymDescTabularHeader</code>	<code>\glslongtok</code>
..... 474, 475	. 127, 218–220, 240, 242–245, 247, 248,
<code>\glslongextraSetDescWidth</code>	251, 253, 256, 259–264, 268, 271, 273,
..... 464–466, 468, 469	277–281, 284, 288, 290, 293–297, 299–
<code>\glslongextraSubDescFmt</code> 466, 468–	301, 305, 307, 310, 312, 314, 315, 319–
470, 472, 474, 475, 477, 478, 480, 481, 483	327, 329–331, 333, 335, 337–340, 343–345
<code>\glslongextraSubLocationFmt</code>	<code>\glslonguserfont</code> 318, 319, 321, 323, 324, 326
..... 468, 470, 474, 477, 480, 483	<code>\glsmcols</code> 448, 450, 451, 453
<code>\glslongextraSubNameFmt</code> 466, 468–	<code>\GLSname</code> 352
470, 472, 473, 475, 477, 478, 480, 481, 483	<code>\Glsname</code> 352
<code>\glslongextraSubSymbolFmt</code>	<code>\glsname</code> 352
.... 472, 474, 475, 477, 478, 480, 481, 483	<code>\glsnameaccessdisplay</code> 172, 197–199
<code>\glslongextraSymbolAlign</code> 471–479, 481–483	<code>\glsnamefont</code> 197–200, 202, 464
<code>\glslongextraSymbolFmt</code>	<code>\glsnavhyperlink</code> 142
462, 472, 473, 475, 477, 478, 480, 481, 483	<code>\glsnavhyperlinkname</code> 100
<code>\glslongextraSymDescNameHeader</code> 478	<code>\glsnavhypertarget</code>
<code>\glslongextraSymDescNameTabularFooter</code> 419, 420, 430, 433, 435, 447–453
..... 477, 478	<code>\glsnavigation</code>
<code>\glslongextraSymDescNameTabularHeader</code> 419, 420, 430, 433, 435, 447–453
..... 477, 478	<code>\glsnextpages</code> 136
<code>\glslongextraSymLocSetDescWidth</code> ...	<code>\glsnoidxdisplayloc</code> 139, 140, 374
..... 473, 476, 479, 482	<code>\glsnoidxdisplaylocclsthandler</code> 139
<code>\glslongextraSymSetDescWidth</code>	<code>\glsnoidxlocclst</code> 140, 161
..... 465, 471, 474, 475, 477, 478, 481	<code>\glsnoidxnumberlistloophandler</code> 140
<code>\glslongextraTabularVAlign</code>	<code>\glsnonextpages</code> 136
..... 465, 467, 468,	<code>\glsnonumberlistfalse</code> 69
470, 471, 473, 474, 476, 477, 479, 481, 482	<code>\glsnonumberlisttrue</code> 70
<code>\glslongextraUpdateWidest</code> 377–379	<code>\glsnopostdotfalse</code> 137
<code>\glslongextraUpdateWidestChild</code> 377–380	<code>\glsnopostdottrue</code> 136
<code>\GlsLongExtraUseTabularfalse</code> 465	<code>\glsnumberlistloop</code> 134
	<code>\glsnumlistlastsep</code> 139, 371

<code>\glsnumlistsep</code>	139, 371	<code>\glsshortpluralaccessdisplay</code> ..	177, 178
<code>\glsopenbrace</code>	58, 148, 149	<code>\glsshorttok</code>	127, 218–220, 240, 242–248, 250, 251, 253, 254, 258– 265, 267, 271, 273, 275, 277–282, 284, 288–290, 292–297, 299–301, 303, 305, 307, 312, 314, 315, 317, 319, 320, 322– 327, 329, 330, 333, 335, 337–340, 342–345
<code>\glsorder</code>	132, 153	<code>\glsshowtargetfont</code>	28
<code>\glspagelistwidth</code>	421, 423, 425, 427, 463, 465	<code>\glsshowtargetouter</code>	28
<code>\glspar</code>	159, 485	<code>\glssubentryitem</code>	155, 417–428, 430, 432, 434, 446, 458, 462
<code>\glspatchLToutput</code>	466, 467, 469–471, 473, 475, 476, 478, 479, 481, 482	<code>\glssymbolaccessdisplay</code>	174, 175
<code>\glspdffmtfull</code>	367, 368	<code>\glssymbolpluralaccessdisplay</code>	175
<code>\glspdffmtfullpl</code>	368, 369	<code>\glstarget</code>	155, 156, 417–428, 430, 432, 434, 446, 457, 458, 461, 462, 485, 487
<code>\glspenaltygroupskip</code> ...	466, 468, 469, 471, 472, 474, 475, 477, 478, 480, 481, 483	<code>\GLStext</code>	352, 353
<code>\GLSpl</code>	114, 126, 165	<code>\Glstext</code>	353
<code>\Glspl</code>	67, 114, 126, 165	<code>\glstext</code>	353
<code>\glspl</code>	66, 114, 126, 165	<code>\glstextaccessdisplay</code>	172, 173
<code>\GLSplural</code>	353, 354	<code>\glstextformat</code>	75, 78, 79
<code>\Glsplural</code>	354	<code>\glstextup</code>	260
<code>\glsplural</code>	353, 354	<code>\glstopic@postchildren</code>	488, 489
<code>\glspluralaccessdisplay</code>	173	<code>\glstopic@prechildren</code>	484, 488, 489
<code>\glspluralsuffix</code>	152, 218, 219, 223	<code>\glstopic@prevlevel</code>	484, 485, 488, 489
<code>\glspostdescription</code>	19, 20, 136, 137, 211, 417, 418, 420–428, 431–435, 461, 486, 487	<code>\glstopicAssignSubIndent</code>	485, 489
<code>\glspostinline</code>	428	<code>\glstopicAssignWidest</code>	486
<code>\glspostlinkhook</code>	73–75, 89–93, 106, 223–234	<code>\glstopicCols</code>	488, 489
<code>\glsprestandardsort</code>	134	<code>\glstopicColsEnv</code>	488, 489
<code>\glsresetentrylist</code>	142, 157	<code>\glstopicDesc</code>	485
<code>\glssee</code>	58, 60	<code>\glstopicGroupHeading</code>	484, 488
<code>\glsseeformat</code>	54, 57, 63, 133, 139, 140	<code>\glstopicInit</code>	484, 488
<code>\glsseelist</code>	57	<code>\glstopicItem</code>	484, 488
<code>\glssetabbrvfmt</code>	72, 81, 104, 105, 195–201, 207, 208, 223–225, 227, 228, 230–232, 234	<code>\glstopicLoc</code>	485
<code>\glssetattribute</code>	241–244, 246, 248, 249, 251, 253–260, 262, 264, 266, 268, 271, 273, 274, 276–282, 284, 288, 290–292, 294– 299, 301, 303, 305, 307, 308, 310, 312– 315, 317, 319, 320, 322, 323, 325–327, 329–331, 333, 335, 337–339, 341, 343–345	<code>\glstopicMarker</code>	485
<code>\glssetcategoryattribute</code>	115, 126, 130, 165, 171, 184, 185, 190, 191, 193, 194, 209	<code>\glstopicMidSkip</code>	485
<code>\glssetnoexpandfield</code>	14	<code>\glstopicParIndent</code>	484, 488
<code>\glssettocitle</code>	135	<code>\glstopicPostSkip</code>	485
<code>\glssetwidest</code>	378	<code>\glstopicPreSkip</code>	485
<code>\glsshortaccessdisplay</code>	177	<code>\glstopicSubIndent</code>	486
<code>\glsshortaccsupp</code>	184	<code>\glstopicSubItem</code>	485, 489
<code>\glsshortpltok</code>	220, 240, 242–248, 250, 251, 253, 254, 260–266, 271, 273, 275, 277– 282, 288, 290, 292, 294–297, 299, 301, 303, 312, 314, 315, 317, 319, 320, 322– 327, 329, 330, 335, 337–340, 342, 344, 345	<code>\glstopicSubItemBox</code>	487
		<code>\glstopicSubItemSep</code>	487
		<code>\glstopicSubLoc</code>	487
		<code>\glstopicSubNameFont</code>	487
		<code>\glstopicSubPreLocSep</code>	487
		<code>\glstopicTitle</code>	485
		<code>\glstopicTitleFont</code>	485
		<code>\glstopicwidest</code>	486, 487
		<code>\glstreechilddesc</code>	432

<code>\glstreeChildDescLoc</code>	430, 432	<code>\glxtr@fields</code>	152
<code>\glstreechildpredesc</code>	432	<code>\glxtr@float</code>	18
<code>\glstreechildprelocation</code>	432, 434	<code>\glxtr@grptitle</code> ...	430, 433–435, 447–453
<code>\glstreechildsymbol</code>	430, 432	<code>\glxtr@headentry@p</code>	108, 109
<code>\glstreedefaultnamefmt</code>	429	<code>\glxtr@hyperoutsidefalse</code>	76
<code>\glstreedesc</code>	431	<code>\glxtr@hyperoutsidettrue</code>	75, 76
<code>\glstreeDescLoc</code>	430, 432, 434, 435	<code>\glxtr@ifnextpunc</code>	215
<code>\glstreegroupheaderfmt</code>		<code>\glxtr@ifpunctoken</code>	215
.....	430, 433–435, 447–453, 455	<code>\glxtr@inc@linkcount</code>	78, 171
<code>\glstreegroupheaderskip</code>		<code>\glxtr@inc@wrglossaryctr</code>	8, 9, 26, 30
.....	430, 431, 433–435, 447–453	<code>\glxtr@indexonly@saveentrycounter</code>	
<code>\glstreegroupskip</code> ..	429, 430, 432, 434, 446	16, 30
<code>\glstreeindent</code>	432, 434, 445, 446	<code>\glxtr@keylist</code>	65–67
<code>\glstreeitem</code>	429, 449, 457, 458	<code>\glxtr@label</code>	460
<code>\glstreenamebox</code>	446	<code>\glxtr@langtag</code>	152
<code>\glstreenamefmt</code>		<code>\glxtr@linkprefix</code>	152, 153
.....	428, 430, 432, 434, 436, 438–446	<code>\glxtr@loaddialect</code>	370, 415
<code>\glstreenavigationfmt</code>	430, 433, 435, 447–453	<code>\glxtr@locationhypertext</code>	376
<code>\glstreeNoDescSymbolPreLocation</code>	431, 432	<code>\glxtr@makeglossaries</code>	132
<code>\glstreenonamechilddesc</code>	434	<code>\glxtr@newabbreviation</code>	129, 218
<code>\glstreenonamedesc</code>	434	<code>\glxtr@next</code>	215
<code>\glstreenonamesymbol</code>	434	<code>\glxtr@org@@do@wrglossary</code>	30
<code>\glstreepredesc</code>	431, 433	<code>\glxtr@org@dohyperlink</code>	100
<code>\glstreePreHeader</code> ..	430, 433–435, 447–453	<code>\glxtr@org@getgrouptitle</code>	141
<code>\glstreeprelocation</code>	429, 431	<code>\glxtr@org@newignoredglossary</code>	49
<code>\glstreesubitem</code>	429, 458	<code>\glxtr@orgmakenoidxglossaries</code>	63
<code>\glstreesubsubitem</code>	429, 458	<code>\glxtr@pluralsuffixes</code>	152
<code>\glstreesymbol</code>	430, 432	<code>\glxtr@process</code>	157, 158
<code>\GlsrLetField</code>	41	<code>\glxtr@provideignoredglossary</code>	50
<code>\glstype</code>	74, 77, 89–93, 166, 223–234	<code>\glxtr@punclist</code>	214, 215
<code>\glunset</code>	62, 74, 117–119, 166	<code>\glxtr@record</code>	11, 12, 152
<code>\glupdatewidest</code>	377, 378	<code>\glxtr@record@nameref</code>	12, 152
<code>\gluserdescription</code>	319, 320, 323, 326	<code>\glxtr@record@nr</code>	15, 16
<code>\glswrite</code>	58, 132	<code>\glxtr@recordsee</code>	13
<code>\glswriteentry</code>	8, 9	<code>\glxtr@renewcommand</code>	372
<code>\Glsxtr</code>	67	<code>\glxtr@resource</code>	150, 152
<code>\glxtr</code>	67	<code>\glxtr@s@newignoredglossary</code>	49
<code>\glxtr@@do@wrglossary</code>	8, 10, 13, 16	<code>\glxtr@s@provideignoredglossary</code> ...	50
<code>\glxtr@addloclistfield</code>	15, 16	<code>\glxtr@saveentrycounter</code>	8, 10, 13, 97
<code>\glxtr@addunused</code>	61	<code>\glxtr@setaccessdisplay</code>	201
<code>\glxtr@applyabbrvfmt</code>	234	<code>\glxtr@setbookindexmark</code>	459
<code>\glxtr@applyabbrvstyle</code>	218, 237	<code>\glxtr@setup@record</code>	15, 16, 30, 31
<code>\glxtr@beginbookindex</code>	456	<code>\glxtr@shortcutsval</code>	152, 153
<code>\glxtr@counterrecord</code>	154	<code>\glxtr@texencoding</code>	153
<code>\glxtr@dblfloat</code>	18	<code>\glxtr@undefaction@nr</code>	7
<code>\glxtr@do@alsoindex@wrglossary</code>	16	<code>\glxtr@undefaction@val</code>	7
<code>\glxtr@do@autoadd</code>	9, 77, 78	<code>\glxtr@usesee</code>	54
<code>\glxtr@doooption</code>	5, 18, 19, 26, 27, 30	<code>\glxtr@warnonexistsordo</code>	7, 15, 16, 53
<code>\glxtr@endbookindex</code>	457	<code>\glxtr@writefields</code>	150

<code>\glxtrabbreviationfont</code>	72	<code>\glxtrbibnote</code>	373
<code>\glxtrabbrvfootnote</code> 245–247, 249, 250,		<code>\glxtrbibnumber</code>	373
271–274, 276, 288–290, 292, 312–315, 317		<code>\glxtrbiborganization</code>	373
<code>\glxtrabbrvpluralsuffix</code> ...	152, 182,	<code>\glxtrbibpages</code>	373
223, 241, 243, 246, 249, 252, 254, 257,		<code>\glxtrbibpublisher</code>	374
260, 276, 293, 318, 329, 332, 335, 341, 343		<code>\glxtrbibschool</code>	374
<code>\glxtrabbrvtype</code>	20, 220	<code>\glxtrbibseries</code>	374
<code>\glxtrAccSuppAbbrSetFirstLongAttrs</code>		<code>\glxtrbibtitle</code>	374
.....	240, 243,	<code>\glxtrbibtype</code>	374
260, 262, 277, 279, 293, 295, 297, 299,		<code>\glxtrbibvolume</code>	374
319, 320, 322, 326, 329, 335, 338, 340, 343		<code>\glxtrbookindexatendgroup</code>	457
<code>\glxtrAccSuppAbbrSetNameLongAttrs</code>		<code>\glxtrbookindexatsubendgroup</code>	458
.....	247,	<code>\glxtrbookindexatsubsubendgroup</code> ..	458
250, 265, 273, 275, 280, 289, 292, 314, 317		<code>\glxtrbookindexbetween</code>	457
<code>\glxtrAccSuppAbbrSetNameShortAttrs</code>		<code>\glxtrbookindexbookmark</code>	459
.....	259, 284, 305, 306, 308, 333	<code>\glxtrbookindexbookmarkprefix</code> ...	459
<code>\glxtrAccSuppAbbrSetNoLongAttrs</code> ..		<code>\glxtrbookindexcols</code>	456, 457
.....	245, 248, 251, 253, 264,	<code>\glxtrbookindexcolspread</code>	456
271, 273, 282, 288, 290, 301, 303, 312, 315		<code>\glxtrbookindexfirstmarkfmt</code>	460
<code>\glxtrAccSuppAbbrSetTextShortAttrs</code>		<code>\glxtrbookindexformatheader</code>	459
.....	242, 244,	<code>\glxtrbookindexgroupskip</code>	459
261, 263, 278, 280, 295, 296, 298, 300,		<code>\glxtrbookindexlastmarkfmt</code>	460
322, 324, 325, 327, 330, 336, 339, 342, 345		<code>\glxtrbookindexlocation</code>	455, 457
<code>\glxtractivatenopost</code>	136	<code>\glxtrbookindexmulticolenv</code> ..	456, 457
<code>\glxtraddallcrossrefs</code>	61	<code>\glxtrbookindexname</code>	454, 457
<code>\glxtralias</code>	97	<code>\glxtrbookindexparentchildsep</code>	455–457
<code>\glxtrAltTreeIndent</code>	435, 436	<code>\glxtrbookindexparentschildsep</code> .	
<code>\glxtralttreeInit</code>	445, 451, 453	456–458
<code>\glxtrAltTreePar</code>	435	<code>\glxtrbookindexprelocation</code> ...	454, 457
<code>\glxtrAltTreeSetHangIndent</code> ...	435, 446	<code>\glxtrbookindexsubbetween</code>	458
<code>\glxtrAltTreeSetSubHangIndent</code> ...	446	<code>\glxtrbookindexsublocation</code>	458
<code>\glxtralttreeSubSymbolDescLocation</code>	446	<code>\glxtrbookindexsubname</code>	458
<code>\glxtralttreeSymbolDescLocation</code> ..		<code>\glxtrbookindexsubprelocation</code> ...	458
.....	436, 446	<code>\glxtrbookindexsubsubbetween</code>	458
<code>\glxtrassignactualsetup</code>	181	<code>\glxtrbookindexthepage</code>	459, 460
<code>\glxtrassignfieldfont</code>	81–88	<code>\glxtrcat</code>	66, 67
<code>\glxtrautoindex</code>	204	<code>\glxtrchecknohyperfirst</code>	82–84
<code>\glxtrautoindexassignsort</code>	204	<code>\glxtrcombiningdiacriticIIIrules</code> .	387
<code>\glxtrautoindexentry</code>	204	<code>\glxtrcombiningdiacriticIIrules</code> ..	387
<code>\glxtrautoindexesc</code>	204	<code>\glxtrcombiningdiacriticIrules</code> ...	387
<code>\glxtrbibaddress</code>	373	<code>\glxtrcombiningdiacriticIVrules</code> ..	387
<code>\glxtrbibauthor</code>	373	<code>\glxtrComputeTreeIndent</code>	446
<code>\glxtrbibbooktitle</code>	373	<code>\glxtrComputeTreeSubIndent</code>	446
<code>\glxtrbibchapter</code>	373	<code>\glxtrcounterprefix</code>	144
<code>\glxtrbibedition</code>	373	<code>\glxtrcurrencyrules</code>	389
<code>\glxtrbibhowpublished</code>	373	<code>\glxtrcurrentgrptitle</code>	459
<code>\glxtrbibinstitution</code>	373	<code>\GlsXtrDefaultResourceOptions</code>	150
<code>\glxtrbibjournal</code>	373	<code>\GlsXtrdefaultsubsequentfmt</code> ...	236, 238
<code>\glxtrbibmonth</code>	373	<code>\glxtrdefaultsubsequentfmt</code> ...	236, 238

\Glsxtrdefaultsubsequentplfmt .	236, 238	286–288, 291, 294, 296, 298, 300, 302,
\glxtrdefaultsubsequentplfmt .	236, 238	304, 306, 308, 310, 312, 313, 316, 320,
\GlsXtrDefineAbbreviationShortcuts .	23	321, 323, 327, 330, 333, 336, 338, 341, 344
\GlsXtrDefineAcShortcuts	23	\glxtrfullformat
\GlsXtrDefineOtherShortcuts	23 222, 236, 238, 241, 243, 246,
\glxtrdetoklocation	164	249, 252, 254, 258, 261, 263, 265, 266,
\glxtrdiscardperiod	212	269–271, 274, 277, 279, 281, 283, 285,
\glxtrdisplayendloc	143	287, 288, 291, 294, 296, 298, 300, 302,
\glxtrdisplayendlohook	144	303, 306, 308, 310, 311, 313, 316, 319,
\glxtrdisplaysingleloc	143, 144	321, 323, 326, 330, 332, 336, 338, 341, 344
\glxtrdisplaystartloc	143	\GLSxtrfullpl
\glxtrdoautoindexname	98, 99, 200	21, 22, 94, 358–360
\glxtrdohyperlink	100, 102, 103	\Glsxtrfullpl
\glxtrdopostpunc		21, 93, 360
....	249, 250, 274, 276, 290, 292, 315, 317	\glxtrfullpl
\glxtrdowrglossaryhook	98, 99	222, 235, 238, 241, 244, 246, 249,
\GlsXtrDualField	373	252, 254, 258, 261, 263, 265, 266, 269,
\glxtrdmsuffix	294, 296, 298,	270, 272, 274, 278, 280, 282, 283, 286,
	300, 301, 303, 305, 307, 309, 310, 313, 316	287, 289, 291, 294, 296, 298, 300, 302,
\GlsXtrEnableEntryCounting	126	304, 306, 308, 310, 312, 313, 316, 320,
\GlsXtrEnableEntryUnitCounting	114, 115	321, 323, 327, 330, 333, 336, 338, 341, 344
\GlsXtrEnableOnTheFly	65, 67, 68	\glxtrfullplformat
\glxtrendfor	37 235, 238, 241, 243, 246,
\glxtrfieldlistgadd	154	249, 252, 254, 258, 261, 263, 265, 266,
\glxtrfieldtitlecase	195–198, 202	269, 270, 272, 274, 277, 279, 282, 283,
\glxtrfieldtitlecasecs	194	286–288, 291, 294, 296, 298, 300, 302,
\glxtrfieldxifinlist	159	304, 306, 308, 310, 311, 313, 316, 319,
\glxtrfirstscfont	260	321, 323, 327, 330, 333, 336, 338, 341, 344
\glxtrfirstsmfont	276	\glxtrfullsep
\GlsXtrFmtDefaultOptions	34 221, 240–244, 247, 249, 250, 252–
\glxtrfmtdisplay	34, 35	257, 260–270, 272, 275, 277–285, 287,
\glxtrfmtexternalnameref	375, 376	289, 291–304, 306–311, 313, 314, 316,
\GlsXtrFmtField	34, 35	318, 328–330, 332, 334, 337–340, 344, 345
\glxtrfmtinternalnameref	375, 376	\glxtrrgenabbrvfmt
\glxtrfootnotedescname		72
....	247, 250, 273, 275, 289, 292, 314, 317	\glxtrrgeneralpuncIIrules
\glxtrfootnotedescsort		389
....	247, 250, 273, 275, 289, 292, 314, 317	\glxtrrgeneralpuncIrules
\glxtrfootnotename		389
....	245, 248, 271, 273, 288, 290, 312, 315	\glxtrrgetgrouptitle
\GlsXtrForeignTextField	44 142, 430, 433–435, 447–453, 459
\GlsXtrFormatLocationList	69, 72, 443, 444	\glxtrrheadfirst
\GLSxtrfull	21, 93, 358, 359	348
\Glsxtrfull	21, 93, 359	\glxtrrheadfirst
\glxtrfull	20, 21, 93, 358	348
\Glsxtrfullformat		\Glsxtrrheadfirstplural
.....	222, 236, 238, 241, 243, 246,	349
	249, 252, 254, 258, 261, 263, 265, 266,	\glxtrrheadfirstplural
	269, 270, 272, 274, 277, 279, 282, 283,	349
		\Glsxtrrheadfull
		349
		\glxtrrheadfull
		349
		\Glsxtrrheadfullpl
		349
		\glxtrrheadfullpl
		349
		\Glsxtrrheadlong
		349
		\glxtrrheadlong
		349
		\Glsxtrrheadlongpl
		349

<code>\glxtrheadlongpl</code>	349	<code>\Glsxtrinlinefullplformat</code> ..	222, 226,
<code>\Glsxtrheadname</code>	348		238, 247, 250, 252, 254, 256, 257, 265–
<code>\glxtrheadname</code>	154, 155, 348		267, 269, 270, 272, 275, 281, 283–285,
<code>\Glsxtrheadplural</code>	348		287, 289, 292, 302–304, 306, 308, 310,
<code>\glxtrheadplural</code>	348		311, 314, 316, 321, 324, 332, 336, 342, 345
<code>\Glsxtrheadshort</code>	348	<code>\glxtrinlinefullplformat</code>	222, 225, 226,
<code>\glxtrheadshort</code>	348		238, 247, 250, 252, 254, 255, 257, 264,
<code>\Glsxtrheadshortpl</code>	348		266–268, 270, 272, 275, 281, 283–285,
<code>\glxtrheadshortpl</code>	348		287, 289, 291, 302–304, 306, 307, 309,
<code>\Glsxtrheadtext</code>	348		311, 314, 316, 321, 324, 332, 336, 342, 344
<code>\glxtrheadtext</code>	348	<code>\glxtrinsertinsidefalse</code>	240
<code>\glxtrhiernamesep</code>	55, 56	<code>\GlsXtrInternalLocationHyperlink</code>	26, 145
<code>\glxtrhyperlink</code>	26, 101	<code>\glxtrLatinA</code>	391–396
<code>\glxtrhyphensuffix</code>	330, 338	<code>\glxtrLatinAELigature</code>	393, 395, 396
<code>\glxtridentifyglsl</code>	162	<code>\glxtrLatinE</code>	391–396
<code>\glxtrifcounttrigger</code>	117–119	<code>\glxtrLatinEszettSs</code>	392–394, 396
<code>\glxtrifcustomdiscardperiod</code>	212	<code>\glxtrLatinEszettSz</code>	393, 395
<code>\glxtrifemptyglossary</code>	142, 149, 157	<code>\glxtrLatinEth</code>	392–395
<code>\GlsXtrIfFieldEqNum</code>	155	<code>\glxtrLatinH</code>	391–396
<code>\GlsXtrIfFieldNonZero</code>	371	<code>\glxtrLatinI</code>	391–396
<code>\glxtrifhasfield</code> 44, 55, 56, 97, 372, 373, 454		<code>\glxtrLatinInsularG</code>	395
<code>\glxtrifhyphenstart</code> 328, 331, 334, 337, 340		<code>\glxtrLatinK</code>	391–396
<code>\glxtrifindexing</code>	98	<code>\glxtrLatinL</code>	391–396
<code>\glxtrifinmark</code>	80, 108–111, 347–349	<code>\glxtrLatinM</code>	391–396
<code>\glxtrifnextpunc</code>	215, 216	<code>\glxtrLatinN</code>	391–396
<code>\glxtrifperiod</code>	212, 214	<code>\glxtrLatinO</code>	391–396
<code>\glxtrifrecordtrigger</code>	167–169	<code>\glxtrLatinOELigature</code>	393, 396
<code>\glxtrifwasfirstuse</code>		<code>\glxtrLatinP</code>	391–396
	81–84, 89–93, 96, 130, 213, 224, 227–	<code>\glxtrLatinS</code>	391–396
	233, 248–251, 274–276, 290–293, 315,	<code>\glxtrLatinT</code>	391–396
	317, 320, 322, 323, 325, 335, 337, 340, 342	<code>\glxtrLatinThorn</code>	396
<code>\glxtrinclinkcounter</code>	170	<code>\glxtrLatinX</code>	392–396
<code>\glxtrinindexaliased</code>	97, 98	<code>\GlsXtrLocationField</code>	161
<code>\glxtrinindexseealso</code>	59, 60	<code>\glxtrlocationhyperlink</code>	144
<code>\glxtrinithyperoutside</code>	77	<code>\glxtrlocrangefmt</code>	143, 144
<code>\glxtrinitwrgloss</code>	77, 166	<code>\GLSxtrlong</code>	21, 91, 356, 357
<code>\glxtrinitwrglossbeforefalse</code>	75	<code>\Glsxtrlong</code>	21, 91, 357
<code>\glxtrinitwrglossbeforetrue</code>	75	<code>\glxtrlong</code>	20, 21, 91, 356
<code>\Glsxtrinlinefullformat</code>	222,	<code>\glxtrlonghyphen</code>	336
	224, 238, 247, 250, 252, 254, 255, 257,	<code>\glxtrlonghyphennoshort</code>	332, 333
	265–268, 270, 272, 275, 281, 283–285,	<code>\glxtrlonghyphenshort</code>	330
	287, 289, 292, 302–304, 306, 308, 309,	<code>\glxtrlongnoshortdescname</code> .	256, 310, 331
	311, 314, 316, 321, 324, 332, 336, 342, 345	<code>\glxtrlongnoshortname</code>	
<code>\glxtrinlinefullformat</code> 222–224, 238,			259, 267, 284, 305, 307, 333
246, 249, 252, 254, 255, 257, 264, 266–		<code>\GLSxtrlongpl</code>	21, 92, 357, 358
268, 270, 272, 275, 281, 282, 284, 285,		<code>\Glsxtrlongpl</code>	21, 92, 358
287, 289, 291, 301, 303–305, 307, 309,		<code>\glxtrlongpl</code>	20, 21, 92, 357
311, 313, 316, 321, 324, 332, 336, 341, 344		<code>\glxtrlongshortdescname</code>	
			242, 261, 278, 295, 297, 330, 336

<code>\glxstrlongshortdescsort</code>	<code>\glxstronlyname</code>	343
.... 242, 261, 278, 295, 297, 325, 330, 336	<code>\glxstronlysuffix</code>	344
<code>\glxstrlongshortname</code>	<code>\glxstrorg@ifKV@glslink@hyper</code>	73
240, 260, 277, 293, 295, 319, 320, 329, 335	<code>\glxstrorglong</code>	181, 218, 241, 331
<code>\glxstrlongshortuserdescname</code> ..	<code>\glxstrorgshort</code>	181, 218, 241
322, 325	<code>\GLSxtrp</code>	108
<code>\glxstrmarkhook</code>	<code>\Glsxtrp</code>	107
346, 347	<code>\glxstrp</code>	107, 109
<code>\glxstrMathItalicAlpha</code>	<code>\glxstrparen</code>	213,
401, 404, 405	221, 240–244, 247, 250, 252–257, 260–	
<code>\glxstrMathItalicBeta</code>	270, 272, 275, 277–281, 283–285, 287,	
401, 404, 405	289, 291–304, 306–311, 313, 314, 316,	
<code>\glxstrMathItalicChi</code> ...	318, 328–330, 332, 334, 337–340, 344, 345	
401, 402, 405, 406	<code>\glxstrpdfentryfmt</code>	35
<code>\glxstrMathItalicDelta</code>	<code>\Glsxtrpl</code>	67
401, 404, 406	<code>\glxstrpl</code>	67
<code>\glxstrMathItalicEpsilon</code> ...	<code>\glxstrpostdescription</code> .	137, 193, 211, 428
401, 404, 406	<code>\glxstrposthyphenlong</code>	340, 342
<code>\glxstrMathItalicEta</code>	<code>\glxstrposthyphenshort</code>	335, 337
401, 405, 406	<code>\glxstrposthyphensequent</code>	
<code>\glxstrMathItalicGamma</code>	335, 337, 341, 342	
401, 404, 405	<code>\glxstrpostlink</code>	212
<code>\glxstrMathItalicIota</code>	<code>\glxstrpostlinkendsentence</code>	212
401, 405, 406	<code>\glxstrpostlinkhook</code>	212
<code>\glxstrMathItalicKappa</code>	<code>\glxstrpostlocalreset</code>	113, 116, 124
401, 405, 406	<code>\glxstrpostlocalunset</code>	113, 116, 124
<code>\glxstrMathItalicLambda</code> ...	<code>\glxstrpostlongdescription</code>	48
401, 405, 406	<code>\glxstrpostnamehook</code>	198–200, 202
<code>\glxstrMathItalicMu</code>	<code>\GlsXtrPostNewAbbreviation</code>	
401, 405, 406 221, 238, 241–244, 246, 248, 250,	
<code>\glxstrMathItalicNu</code>	251, 253–264, 266, 268, 271, 273–275,	
401, 405, 406	277–282, 284, 288, 290, 292, 294–297,	
<code>\glxstrMathItalicOmega</code> .	299, 301, 303, 305, 307, 308, 310, 312–	
401, 402, 405, 406	315, 317, 319, 320, 322, 323, 325–327,	
<code>\glxstrMathItalicOmicron</code> ...	329–331, 333, 335, 337–340, 342, 344, 345	
401, 405, 406	<code>\glxstrpostreset</code>	113, 116, 124
<code>\glxstrMathItalicPhi</code> ...	<code>\glxstrpostunset</code>	112, 115, 123, 124
401, 402, 405, 406	<code>\glxstrprelocation</code>	
<code>\glxstrMathItalicPi</code> 418, 420–422, 424, 426, 429, 454	
401, 405, 406	<code>\glxstrprotectlinks</code>	101, 102
<code>\glxstrMathItalicPsi</code> ...	<code>\glxstrprovideaccsuppcmd</code>	184, 185
401, 402, 405, 406	<code>\GlsXtrRecordCounter</code>	13
<code>\glxstrMathItalicRho</code>	<code>\glxstrrecordtriggervalue</code>	165
401, 405, 406	<code>\GlsXtrRecordWarning</code>	150
<code>\glxstrMathItalicSigma</code>	<code>\glxstrregularfont</code>	72, 81
401, 405, 406	<code>\glxstrresourcecount</code>	151
<code>\glxstrMathItalicTau</code>	<code>\glxstrresourcefile</code>	151
401, 405, 406	<code>\glxstrresourceinit</code>	150
<code>\glxstrMathItalicTheta</code>	<code>\glxstrrestoremarkhook</code>	346, 347
401, 405, 406		
<code>\glxstrMathItalicUpsilon</code> ...		
401, 405, 406		
<code>\glxstrMathItalicXi</code>		
401, 405, 406		
<code>\glxstrMathItalicZeta</code>		
401, 405, 406		
<code>\glxstrmultisupplocation</code>		374
<code>\glxstrnamereflink</code>		375
<code>\glxstrnewabbrevpresetkeyhook</code>		219
<code>\glxstrnewnumber</code>		22
<code>\glxstrnewsymbol</code>		22
<code>\glxstrNoGlossaryWarning</code>		15, 24, 145
<code>\GlsXtrNoGlsWarningAutoMake</code>		149
<code>\GlsXtrNoGlsWarningBuildInfo</code>		149
<code>\GlsXtrNoGlsWarningCheckFile</code>		149
<code>\GlsXtrNoGlsWarningEmptyMain</code>		149
<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...		149
<code>\GlsXtrNoGlsWarningEmptyStart</code>		149
<code>\GlsXtrNoGlsWarningHead</code>		149
<code>\GlsXtrNoGlsWarningMisMatch</code>		149
<code>\GlsXtrNoGlsWarningNoOut</code>		149
<code>\GlsXtrNoGlsWarningTail</code>		150
<code>\glxstrnopostpunc</code>		136
<code>\glxstronlydescname</code>		345
<code>\glxstronlydescsort</code>		345

\glsxtrrestorepostpunc	137	\glsxtrsubsequentplfmt	
\glsxtrscfont	260 234, 238, 257, 268, 270,	
\glsxtrscsuffix		285, 286, 305, 307, 309, 311, 332, 335, 341	
.... 260, 262, 264, 266, 268, 269, 271, 274		\glsxtrsupplocationurl	145, 374, 376
\GlsXtrSetActualChar	206	\glsxtrtagfont	210
\glsxtrsetaliasnoindex	15, 16, 97, 98	\GLSxtrtitlefirst	364
\GlsXtrSetEncapChar	207	\Glsxtrtitlefirst	348, 349, 364
\GlsXtrSetEscChar	206	\glsxtrtitlefirst	348, 349, 364
\glsxtrsetfieldifexists	41, 42	\GLSxtrtitlefirstplural	365
\glsxtrsetglossarylabel	138	\Glsxtrtitlefirstplural	348, 349, 365
\GlsXtrSetLevelChar	207	\glsxtrtitlefirstplural	348, 349, 365
\glsxtrsetpopts	106	\GLSxtrtitlefull	368
\glsxtrsetupfulldefs	223–	\Glsxtrtitlefull	348, 349, 368
226, 249, 251, 274, 276, 291, 293, 315, 317		\glsxtrtitlefull	348, 349, 367
\GLSxtrshort	21, 89, 110, 111, 350, 351	\GLSxtrtitlefullpl	369
\Glsxtrshort	21, 89, 351	\Glsxtrtitlefullpl	348, 349, 368, 369
\glsxtrshort	20, 21, 88, 350	\glsxtrtitlefullpl	348, 349, 368
\glsxtrshortdescname ...	253, 265, 282, 303	\GLSxtrtitlelong	366
\glsxtrshorthyphen	341	\Glsxtrtitlelong	348, 349, 366
\glsxtrshorthyphenlong	338, 339	\glsxtrtitlelong	348, 349, 365, 366
\glsxtrshortlongdescname		\GLSxtrtitlelongpl	367
..... 244, 263, 280, 298, 300, 339, 342		\Glsxtrtitlelongpl	348, 349, 367
\glsxtrshortlongdescsort		\glsxtrtitlelongpl	348, 349, 366
.... 244, 263, 280, 298, 300, 327, 339, 342		\GLSxtrtitlename	362
\glsxtrshortlongname		\Glsxtrtitlename	348, 349, 361, 362
243, 262, 279, 297, 299, 323, 326, 338, 340		\glsxtrtitlename	348, 349, 361
\glsxtrshortlonguserdescname ..	324, 327	\glsxtrtitleorpdforheading	
\glsxtrshortnolongname .	251, 264, 281, 301 28, 154–156, 348, 349	
\GLSxtrshorttpl	21, 90, 350, 351	\GLSxtrtitleplural	363, 364
\Glsxtrshorttpl	21, 90, 351	\Glsxtrtitleplural	348, 349, 363
\glsxtrshorttpl	20, 21, 90, 350	\glsxtrtitleplural	348, 349, 363
\glsxtrsmfont	276	\Glsxtrtitleshort	348, 349, 361
\glsxtrsmsuffix		\glsxtrtitleshort	348, 349, 360
.... 277, 279, 281, 282, 285, 286, 288, 291		\Glsxtrtitleshortpl	348, 349, 361
\GlsXtrStandaloneEntryName	155	\glsxtrtitleshortpl	348, 349, 360
\GlsXtrStandaloneEntryOther	156	\GLSxtrtitletext	363
\GlsXtrStandaloneGlossaryType .	155, 156	\Glsxtrtitletext	348, 349, 362
\GlsXtrStandaloneSubEntryItem .	155, 156	\glsxtrtitletext	348, 349, 362
\Glsxtrsubsequentfmt		\GlsXtrTotalRecordCount	165
..... 235, 238, 257, 268, 270,		\glsxtrtreetopindent	436, 445
285, 286, 305, 307, 309, 311, 332, 335, 341		\glsxtrundefaction ..	7, 15, 16, 32, 49, 52, 53
\glsxtrsubsequentfmt		\glsxtrundeftag	31, 139, 140
..... 235, 238, 257, 268, 269,		\GlsXtrUnknownDialectWarning	45
285, 286, 305, 307, 309, 311, 332, 335, 341		\glsxtrunsrtdo	158, 159
\Glsxtrsubsequentplfmt		\glsxtrUpAlpha	399, 400, 404, 405
..... 234, 238, 257, 268, 270,		\glsxtrUpBeta	399, 400, 404, 405
285, 286, 305, 307, 309, 311, 332, 335, 341		\glsxtrUpChi	400, 405, 406
		\glsxtrUpDelta	399, 400, 404, 406
		\glsxtrUpDigamma	399, 401, 405

[illegible]

<code>\ifglentryexists</code>	9, 32, 52, 53, 65–67, 69, 70, 81, 161, 190, 191, 211, 212, 381
<code>\ifglshasdesc</code>	189
<code>\ifglshasfield</code>	34, 35, 318
<code>\ifglshaslong</code>	120, 169
<code>\ifglshasparent</code>	155, 156, 158, 161, 438–440
<code>\ifglshasshort</code>	54–56, 72, 81
<code>\ifglshassymbol</code>	213, 431–433, 435, 485, 487
<code>\ifglindexonlyfirst</code>	98
<code>\ifGlsLongExtraUseTabular</code>	465, 467, 468, 470, 471, 473, 474, 476, 477, 479, 480, 482
<code>\ifglsnogroupskip</code>	418, 421–428, 430, 432, 434, 446, 455, 466, 468–470, 472, 474, 475, 477, 478, 480, 481, 483
<code>\ifglsnonumberlist</code>	72
<code>\ifglsnopostdot</code>	19, 136
<code>\ifglssanitizesort</code>	134
<code>\ifglssubentrycounter</code>	46
<code>\ifglused</code>	61, 96, 98, 116, 125, 130, 234, 438, 439, 441, 442, 444
<code>\ifglsxindy</code>	145, 147
<code>\ifglxtr@hyperoutside</code>	79
<code>\ifglxtrinitwrglossbefore</code>	75, 79, 166
<code>\ifglxtrininsertinside</code>	227–234, 236, 241, 243, 244, 246, 247, 249, 250, 252–258, 261, 263–272, 274, 275, 277–289, 291, 292, 294, 296, 298, 300–314, 316, 319–321, 323, 324, 327–329, 331, 334, 336, 337, 340, 342, 344, 345
<code>\ifHy@hyperindex</code>	203
<code>\ifinlist</code>	112
<code>\ifinlistcs</code>	36, 63
<code>\ifinner</code>	28
<code>\ifKV@glslink@hyper</code>	73, 77, 79
<code>\ifKV@glslink@local</code>	74, 166
<code>\ifKV@glslink@noindex</code>	8, 9, 13, 34, 97, 98
<code>\ifmmode</code>	12, 28
<code>\ifnum</code>	17, 39, 40, 62, 117, 125, 126, 141, 151, 165, 377–379, 432, 434, 445, 446, 456, 457, 484, 488, 489
<code>\ifst@rred</code>	12
<code>\ifstrempty</code>	155, 163, 171, 375, 377–381
<code>\ifstrequal</code>	19, 24, 77, 100, 375
<code>\ifthenelse</code>	149, 212
<code>\IfTrackedDialectHasMapping</code>	44
<code>\IfTrackedLanguageFileExists</code>	369
<code>\ifundef</code>	16, 26, 38, 39, 44, 132, 209–211, 374, 415, 464
<code>\ifx</code>	8, 10–12, 15, 31, 58, 68, 69, 78, 129, 131, 135, 136, 138, 143, 144, 151–153, 203–205, 207, 215, 217, 220, 328, 376, 453
<code>\immediate</code>	62, 116, 117, 125, 145, 146, 153
<code>\index</code>	204
<code>\indexspace</code>	418, 429, 455, 459
<code>\input</code>	369
<code>\inputencodingname</code>	152, 153
<code>\InputIfFileExists</code>	62
<code>\istfilename</code>	132
<code>\item</code>	147, 148, 417–420, 429, 430, 448, 449
J	
<code>\jobname</code>	62, 145, 147–151, 153
K	
<code>\key@ifundefined</code>	13, 14, 32, 33, 94, 157, 160
<code>\KV@glslink@hyperfalse</code>	82, 96, 102, 103
<code>\KV@glslink@hypertrue</code>	102
<code>\KV@glslink@noindexfalse</code>	80, 96, 97
<code>\KV@glslink@noindextrue</code>	97, 103
L	
<code>\L</code>	396, 399
<code>\l</code>	399
<code>\label</code>	26, 138
<code>\large</code>	485
<code>\LaTeX</code>	147, 148
<code>\leaders</code>	417
<code>\leavevmode</code>	49, 77
<code>\let</code>	5, 7–11, 13, 15, 16, 18, 20–22, 28–30, 34, 37, 43–45, 48, 62–64, 68, 70, 71, 73, 74, 77–93, 95–100, 102, 103, 106, 112–116, 123–142, 150, 151, 153, 157, 158, 160, 162, 166, 171, 180–182, 195–205, 208–211, 215, 217–220, 223–233, 236, 238, 249, 251, 274, 276, 291, 293, 315, 317, 346–349, 371, 372, 375, 376, 415, 429, 435, 438, 449, 456–459
<code>\letabbreviationstyle</code>	247, 248, 250, 251, 253, 255, 258, 259, 265, 267, 282, 283, 302, 304
<code>\letcs</code>	33, 38, 39, 54, 56, 57, 61, 75, 78, 94, 139–141, 160, 161, 195–202, 208, 460
<code>\levelchar</code>	207
<code>\linewidth</code>	464
<code>\listadd</code>	121
<code>\listbreak</code>	209
<code>\listcsadd</code>	35
<code>\listcseadd</code>	36, 122

<code>\listcsadd</code>	36, 63	<code>\newabbreviationstyle</code> ..	240, 242, 244, 245, 247, 248, 250, 251, 253, 255, 256, 258–265, 267, 269, 271, 273, 275, 277, 278, 280, 282–284, 286, 287, 289, 290, 292–302, 304–306, 308–310, 312, 314, 315, 317, 319, 320, 322, 324–327, 329– 331, 333, 334, 336, 338–340, 342, 343, 345
<code>\listcsxadd</code>	36, 122	<code>\newacronym</code>	127, 129
<code>\listxadd</code>	112	<code>\newacronymhook</code>	127
<code>\loadglsentries</code>	64, 147	<code>\newacronymstyle</code>	128, 130
<code>\long</code>	48	<code>\newcommand</code> 5–15, 17–57, 59, 61, 62, 64–68, 70–72, 75–77, 79–82, 88, 94, 95, 97–99, 101– 103, 106, 108–115, 117, 119–123, 125, 126, 128–131, 135–141, 143–181, 184– 195, 200, 201, 203–218, 221–234, 236– 242, 244, 245, 247, 251, 253, 256, 258– 260, 276, 293, 318, 321, 324, 328, 329, 331, 334, 337, 339, 340, 343, 345, 347– 369, 371–377, 379–382, 386–414, 416, 418, 428, 429, 431–437, 445, 454–456, 459–469, 471, 472, 474–480, 482, 485–488
M		<code>\newcount</code>	151, 162
<code>\m@ne</code>	372	<code>\newcounter</code>	26, 170
<code>\MakeAcronymsAbbreviations</code>	130	<code>\newentry</code>	22
<code>\makeatletter</code>	62, 145, 150, 206	<code>\newglossary</code>	20, 132
<code>\makeatother</code>	205	<code>\newglossaryentry</code> 22, 62, 64, 115, 123, 127, 193, 194, 220
<code>\makebox</code>	417, 446, 487	<code>\newglossaryentry options</code>	
<code>\makefirstuc</code>	210	access	183
<code>makeglossaries</code>	138	alias	18, 53, 57–61
<code>\makeglossaries</code>	131, 146–149, 153, 154	desc	176, 187
<code>makeindex</code>	490	descplural	176, 177, 187
<code>makeindex</code>	16, 131, 490	description	511
<code>\makenoidxglossaries</code>	147	first 100, 173, 174, 186, 240, 354–356, 364, 490	
<code>\MakeTextUppercase</code>	348	firstaccess	183
<code>\MakeUppercase</code>	348, 349	firstplural	174, 186, 240, 355, 365, 490
<code>\marginpar</code>	28	group	160, 161
<code>\markboth</code>	347	loclist	35
<code>\markright</code>	347	long	178, 188, 365
<code>\mathit</code>	384	longplural	179, 188, 366
<code>\mathrm</code>	383–385	name	54, 172, 185, 203, 351, 352, 361
<code>\maxdimen</code>	68, 69	plural	173, 186, 240, 353, 354, 363
<code>\mbox</code>	419, 420, 445	see	17, 18, 29, 53, 54, 58, 61, 64, 132
<code>\medskip</code>	149, 159, 487	seealso	18, 53, 56, 58, 60, 61, 501
<code>\MessageBreak</code>	64, 67, 117, 126, 131, 132, 134, 135, 150, 237	short	54, 177, 187, 216
<code>mfirstuc package</code>	209, 210	shortaccess	181
<code>\mfirstucMakeUppercase</code>	40, 81– 90, 92, 93, 95, 104–106, 108, 110, 111, 119, 120, 128, 169, 172–179, 185–188, 197, 199, 202, 224, 226, 228, 230, 232–236	shortaccessplural	182
<code>\mfu@checkword@arg</code>	209, 210	shortplural	178, 188, 216
<code>\mfu@checkword@do</code>	210		
<code>\midrule</code>	463, 466, 468, 469, 471, 472, 474, 476, 477, 479, 480, 482		
N			
<code>\NeedsTeXFormat</code> ...	5, 370, 416, 454, 461, 484		
<code>\new@atom@glossaryentry</code>	62		
<code>\new@command</code>	372		
<code>\new@glossaryentry</code>	64, 134		
<code>\new@ifnextchar</code>	34, 94, 95, 119, 120, 163, 166–168, 214, 223–233, 381, 382		
<code>\newabbr</code>	21, 22		
<code>\newabbreviation</code>	21, 22		
<code>\newabbreviationhook</code>	220		

shortcuts	22	\printunsrtglossarypredoglossary ..	158
ac	22	\printunsrtglossaryskipentry	157
all	22	\printunsrtglossaryunit	15, 16, 159
false	22	\printunsrtglossaryunitsetup	159
none	22	\ProcessOptions	417
true	22	\ProcessOptionsX	27
sort		\protect	28, 108–111, 185–188, 217, 221, 240, 242–282, 284, 285, 287– 290, 292–297, 299–315, 317, 319–327, 329–333, 335, 337–340, 342–345, 350–359
use	80	\protected@csedef	41, 42, 141, 436, 437
style	25	\protected@csxdef	42, 141, 436, 437
stylemods	25	\protected@edef
symbols	22, 193	... 11, 31, 43, 68, 76, 77, 100, 127, 138, 140, 141, 159, 160, 181, 203, 204, 220, 376	
undefaction	52	\protected@write	11–13, 63, 71, 100, 132, 133, 150, 152–154, 162, 381, 459
error	6	\providecommand	20–22, 33, 57, 71, 96, 97, 100, 116, 125, 131, 132, 145, 146, 152, 162, 370–372, 381, 383–385, 417, 454
warn	6	\ProvidesFile	369
xindy	57, 58	\ProvidesPackage ..	5, 370, 416, 454, 461, 484
\PackageError	6, 13, 25, 29, 62, 64, 67, 68, 76, 94, 95, 98, 100, 107, 115, 116, 123, 125, 126, 130–132, 134, 142, 153, 158, 159, 162, 212, 237–239, 417		
\PackageWarning	18		
\PackageWarningNoLine	18		
\pagelistname ...	466, 469, 472, 476, 479, 482		
\pageref	26, 46		
\par	148–150, 419, 420, 429, 432– 435, 445, 447, 449–453, 455, 484, 486, 488		
\parindent	429, 432, 434– 436, 446–448, 450–453, 456, 484, 486, 488		
\parskip	429, 432, 434, 448, 450, 451, 456		
\PassOptionsToPackage	5, 24		
\pdfbookmark	455, 456		
\pdfstringdef	181		
\preglossarypreamble	47		
\preto	97, 380		
\print@noop@unsrtglossaryunit ...	13, 16		
\print@op@unsrtglossaryunit	15, 16		
\printabbreviations	20		
\printglossaries	133, 147		
\printglossary	20, 133, 147, 148, 150		
\printglossary options			
nonumberlist	71		
type	134		
\printnoidxglossaries	148		
\printnoidxglossary	133, 148		
\printnumbers	22, 194		
\printsymbols	22, 193		
\printunsrtglossary	148, 150, 157, 370, 371		
\printunsrtglossaryentryprocesshook	157, 158		
\printunsrtglossaryhandler	158, 159		
		Q	
		\quad	487
		\quotechar	206
		R	
		\raggedright	422, 423, 426, 427, 457, 463
		\refstepcounter	26
		\relax	7, 15–17, 21, 22, 24, 27, 29, 30, 34, 39, 40, 62, 64, 68, 69, 71, 74, 75, 77, 78, 99, 106, 116, 117, 125, 132, 133, 135, 136, 139–141, 143, 150, 151, 153, 161, 162, 165, 204, 205, 207, 210, 212, 213, 217, 328, 372, 377–379, 430, 432, 434, 438–447, 452, 453, 456–459, 484, 486–489
		resize package	276
		\renewcommand
		. 6, 7, 15–20, 22–31, 47–54, 62–65, 67– 74, 77, 94, 96–98, 100, 102, 106, 113– 116, 120, 123–138, 140, 142–146, 159, 165, 193, 195, 196, 198–200, 207–212, 222, 238, 240–317, 319–327, 329–333, 335–347, 371, 380, 415, 417–430, 432– 435, 445–453, 457, 458, 466–485, 488, 489	
		\renewenvironment	418, 420–427, 429, 432, 434, 445, 448, 450, 451, 453, 456, 465–471, 473–479, 481, 482, 484, 488

<code>\renewglossarystyle</code>	<code>\setglossarystyle</code>
417–427, 429, 430, 432–435, 445, 447–453	25, 135, 417, 419, 420, 430, 433–435, 447–453, 456
<code>\renewrobustcmd</code>	<code>\setkeys</code>
80, 102	9, 25, 30, 34, 78, 80, 98, 127, 136, 166, 218, 220
<code>\RequireGlossariesExtraLang</code> ...	<code>\setlength</code>
370, 415	68, 69, 429, 432, 434, 435, 446, 448, 450, 451, 456, 464, 465, 486, 487
<code>\RequirePackage</code>	<code>\settowidth</code>
5, 15, 24, 25, 27, 416, 454, 461, 484	131, 436–445, 464, 486
<code>\reserved@a</code>	<code>\setupglossaries</code>
215	5, 30
<code>\reserved@b</code>	<code>\sfcode</code>
215	19, 212, 213, 428
<code>\reserved@d</code>	<code>\small</code>
215	28, 56
<code>\RestoreAcronyms</code>	<code>\smallskip</code>
127, 130	487
<code>\rGLS</code>	soul package
165	112
<code>\rGls</code>	<code>\space</code>
165	6, 13, 58, 64, 65, 67, 88, 98, 115–117, 123, 125–127, 130–135, 146, 149, 150, 153, 154, 158, 159, 162, 213, 217, 221, 241, 417, 418, 428, 431, 433, 435, 436, 445, 454, 485, 487, 488
<code>\rgls</code>	<code>\spacefactor</code>
165	19, 212, 213, 428
<code>\rGLSformat</code>	<code>\stepcounter</code>
168	170
<code>\rGlsformat</code>	<code>\string</code>
167	6, 11–13, 31, 58, 63–65, 67, 71, 78, 88, 94, 95, 98, 100, 107, 108, 115–117, 123, 125–127, 130–135, 145–150, 152–154, 158, 159, 162, 197–200, 202, 204, 212, 371, 372, 381, 386–415, 459
<code>\rglsformat</code>	<code>\strut</code>
167, 169	417–428, 434, 462
<code>\rGLSpl</code>	<code>\subglossentry</code> ..
165	136, 161, 417–428, 430, 432, 434, 446, 457, 466, 468–470, 472, 473, 475, 477, 478, 480, 481, 483, 484, 489
<code>\rGlspl</code>	<code>\subitem</code>
165	429, 430
<code>\rglspl</code>	<code>\subsubitem</code>
165	429, 430
<code>\rGLSplformat</code>	<code>\symbolname</code>
169	464, 471, 472, 474, 475, 477, 479, 480, 482
<code>\rGlsplformat</code>	
168	
<code>\rglsplformat</code>	
167, 169	
<code>\robustify</code>	
371	
<code>\romannumeral</code>	
436, 437, 445, 486, 487	
S	
<code>\s@@glsxtrfmt</code>	
34	
<code>\s@gls@hyp@opt</code>	
99	
<code>\s@glsxtr@enabletagging</code>	
209	
<code>\s@glsxtrfmt</code>	
34	
<code>\s@GlsXtrIfFieldCmpNum</code>	
39	
<code>\s@GlsXtrIfFieldEqNum</code>	
39	
<code>\s@GlsXtrIfFieldEqStr</code>	
42	
<code>\s@GlsXtrIfFieldEqXpStr</code>	
42	
<code>\s@GlsXtrIfFieldNonZero</code>	
38	
<code>\s@GlsXtrIfFieldValueInCsvList</code>	
37	
<code>\s@glsxtrifhasfield</code>	
38, 42, 43	
<code>\s@GlsXtrIfXpFieldEqXpStr</code>	
43	
<code>\s@GlsXtrStartUnsetBuffering</code>	
112	
<code>\s@GlsXtrStopUnsetBuffering</code>	
112	
<code>\s@printunsrtglossary</code>	
156, 159	
<code>\seealsoname</code>	
57, 58	
<code>\seename</code>	
54, 57	
<code>\setabbreviationstyle</code>	
130, 241, 253	
<code>\SetAcronymLists</code>	
128–130	
<code>\setacronymstyle</code>	
128–130	
<code>\SetDefaultAcronymDisplayStyle</code>	
130	
<code>\setentrycounter</code>	
143, 374, 376	
<code>\SetGenericNewAcronym</code>	
130	
T	
<code>\tabcolsep</code>	
464, 465	
<code>\tablehead</code>	
424–427	
<code>\tabletail</code>	
424–427	
<code>\tabularnewline</code>	
420–428, 463, 466–483	
<code>\TeX</code>	
147	
<code>\texorpdfstring</code>	
35, 40, 108–110, 208, 347, 360–369	
<code>\textbf</code>	
180, 428, 463, 485, 488	
textcase package	
346	
<code>\textit</code>	
180	
<code>\textmd</code>	
180	
<code>\textrm</code>	
181	
<code>\textsc</code>	
180, 259	
<code>\textsf</code>	
181	
<code>\textsl</code>	
180	

