

# `glossaries-extra.sty v1.16: documented code`

Nicola L.C. Talbot

Dickimaw Books

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

2017-06-15

## Abstract

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

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

# Contents

<b>1</b>	<b>Main Package Code (glossaries-extra.sty)</b>	<b>4</b>
1.1	Package Initialisation and Options . . . . .	4
1.2	Extra Utilities . . . . .	18
1.3	Modifications to Commands Provided by glossaries . . . . .	24
1.3.1	Existence Checks . . . . .	28
1.3.2	Document Definitions . . . . .	35
1.3.3	Existing Glossary Style Modifications . . . . .	40
1.3.4	Entry Formatting, Hyperlinks and Indexing . . . . .	44
1.3.5	Entry Counting . . . . .	77
1.3.6	Acronym Modifications . . . . .	90
1.3.7	Indexing and Displaying Glossaries . . . . .	93
1.4	Integration with glossaries-accsupp . . . . .	115
1.5	Categories . . . . .	126
1.6	Abbreviations . . . . .	149
1.6.1	Abbreviation Styles Setup . . . . .	166
1.6.2	Predefined Styles (Default Font) . . . . .	169
1.6.3	Predefined Styles (Small Capitals) . . . . .	182
1.6.4	Predefined Styles (Fake Small Capitals) . . . . .	186
1.6.5	Predefined Styles (Emphasized) . . . . .	189
1.6.6	Predefined Styles (User Parentheses Hook) . . . . .	196
1.7	Using Entries in Headings . . . . .	204
1.8	Multi-Lingual Support . . . . .	221
<b>2</b>	<b>Style Adjustments (glossaries-extra-stylemods.sty)</b>	<b>223</b>
2.1	Package Initialisation . . . . .	223
2.2	List-Like Styles . . . . .	224
2.3	Longtable Styles . . . . .	224
2.4	Long Ragged Styles . . . . .	225
2.5	Supertabular Styles . . . . .	227
2.6	Super Ragged Styles . . . . .	228
2.7	Inline Style . . . . .	229
2.8	Tree Styles . . . . .	229
	<b>Glossary</b>	<b>241</b>
	<b>Change History</b>	<b>242</b>
	<b>Index</b>	<b>253</b>

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

## 1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2017/06/15 v1.16 (NLCT)]

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

Requires etoolbox package.
4 \RequirePackage{etoolbox}

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

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

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

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

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

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

```

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

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

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

`\glstr@recordsee` Does nothing by default.

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

`@@glstr@record` This is the actual code that does the recording. 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 third argument is the key family (`glslink` in most cases, `glossadd` for `\glsadd`).

```

78 \newcommand*{\@@glstr@record}[3]{%
79   \begingroup
80   \def\@glsnumberformat{glsnumberformat}%
81   \def\@glstr@thevalue{}%
82   \def\@glstr@theHvalue{\@glstr@thevalue}%
83   \ifcsdef{glo@#2@counter}%
84     {%
85       \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
86     }%
87     {%

```

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

```

88     \def\@gls@counter{page}%
89   }%
90   \setkeys{#3}{#1}%

```

```

91 \ifKV@glslink@noindex
92 \else
93 \glswriteentry{#2}%
94 {%

```

Check if the value has been set.

```

95 \ifdefempty{\@glstr@thevalue}%
96 {%

```

Save the entry counter.

```

97 \glstr@saveentrycounter

```

Temporarily redefine \@@do@@wrglossary so we can use \glstr@@do@@wrglossary.

```

98 \let\@@do@@wrglossary\glstr@dorecord
99 }%
100 {%

```

the value has been set, so there's no need to defer writing the location value.

```

101 \let\theHglentrycounter\glstr@thevalue
102 \def\theHglentrycounter{\@glstr@theHvalue}%
103 \let\@@do@@wrglossary\glstr@dorecordnodefer
104 }%
105 \glstr@@do@@wrglossary{#2}%
106 }%
107 \fi
108 \endgroup
109 }

```

glstr@dorecord

```

110 \newcommand*\@glstr@dorecord{%
111 \protected@write\@auxout{\let\@glslocref\relax}{\string\glstr@record
112 {\@gls@label}{\@gls@counterprefix}{\@gls@counter}{\@glsnumberformat}%
113 {\@glslocref}}}%
114 \@glstr@counterrecordhook
115 }

```

dorecordnodefer As above, but don't defer expansion of location.

```

116 \newcommand*\@glstr@dorecordnodefer{%
117 \protected@write\@auxout{}{\string\glstr@record
118 {\@gls@label}{\@gls@counterprefix}{\@gls@counter}{\@glsnumberformat}%
119 {\@glslocref}}}%
120 \@glstr@counterrecordhook
121 }

```

r@recordcounter

```

122 \newcommand*\@@glstr@recordcounter{%
123 \@glstr@noop@recordcounter
124 }

```

p@recordcounter

```

125 \newcommand*{\@glxtr@noop@recordcounter}[1]{%
126   \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
127     requires record=only or record=alsoindex package option}{}%
128 }

p@recordcounter

129 \newcommand*{\@glxtr@op@recordcounter}[1]{%
130   \eappto\@glxtr@counterrecordhook{\noexpand\@glxtr@docounterrecord{#1}}%
131 }

lsxtr@recordsee Deal with \glssee in record mode.
132 \newcommand*{\@glxtr@recordsee}[2]{%
133   \def\@gls@xref{#2}%
134   \@onelevel@sanitize\@gls@xref
135   \protected@write\@auxout{}\string\glxtr@recordsee{#1}{\@gls@xref}}%
136 }

srtglossaryunit

137 \newcommand{\printunsrtglossaryunit}{%
138   \print@noop@unsrtglossaryunit
139 }

tr@setup@record Initialise.
140 \newcommand*{\glxtr@setup@record}{}

aveentrycounter Only store the entry counter information if the indexing is on.
141 \newcommand*{\glxtr@indexonly@saveentrycounter}{%
142   \ifKV@glslink@noindex
143   \else
144     \glxtr@saveentrycounter
145   \fi
146 }

addloclistfield

147 \newcommand*{\glxtr@addloclistfield}{%
148   \key@ifundefined{glossentry}{loclist}%
149   {%
150     \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
151     \appto\@gls@keymap{,{loclist}{loclist}}%
152     \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
153     \appto\@newglossaryentryposthook{%
154       \gls@assign@field{\@glo@label}{loclist}{\@glo@loclist}%
155     }%
156     \glssetnoexpandfield{loclist}%
157   }%
158 }%

```



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

```

159 \key@ifundefined{glossentry}{location}%
160 {%
161   \define@key{glossentry}{location}{\def\@glo@location{##1}}%
162   \appto\@gls@keymap{,{location}{location}}%
163   \appto\@newglossaryentryprehook{\def\@glo@location{}}%
164   \appto\@newglossaryentryposthook{%
165     \gls@assign@field{\@glo@label}{location}{\@glo@location}%
166   }%
167   \glssetnoexpandfield{location}%
168 }%
169 {}%
```

Add a key to store the group heading.

```

170 \key@ifundefined{glossentry}{group}%
171 {%
172   \define@key{glossentry}{group}{\def\@glo@group{##1}}%
173   \appto\@gls@keymap{,{group}{group}}%
174   \appto\@newglossaryentryprehook{\def\@glo@group{}}%
175   \appto\@newglossaryentryposthook{%
176     \gls@assign@field{\@glo@label}{group}{\@glo@group}%
177   }%
178   \glssetnoexpandfield{group}%
179 }%
180 {}%
181 }
```

Now define the record package option.

```

182 \define@choicekey{glossaries-extra.sty}{record}[\val\nr]%
183 {off,only,alsoindex}%
184 [only]%
185 {%
186   \ifcase\nr\relax
```

Don't record.

```

187   \def\glsxtr@setup@record{%
188     \renewcommand*{\@do@seeglossary}{\@glsxtr@org@doseeglossary}%
189     \renewcommand*{\@glsxtr@record}[3]{}%
190     \let\@do@wrglossary\glsxtr@do@wrglossary
191     \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
192     \let\glsxtrundefaction\@glsxtr@err@undefaction
193     \let\glsxtr@warnonexistsordo\@gobble
194     \let\@glsxtr@recordcounter\@glsxtr@noop@recordcounter
195     \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
196     \undef\glsxtrsetaliasnoindex
197   }%
198   \or
```

Only record (don't index).

```

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

```

200 \glsxtr@autoseeindexfalse
201 \let\@do@seeglossary\glsxtr@recordsee
202 \let\glsxtr@record\@glsxtr@record
203 \let\@do@wrglossary\gobble
204 \let\gls@saveentrycounter\relax
205 \let\glsxtrundefaction\glsxtr@warn@undefaction
206 \let\glsxtr@warnonexistssordo\glsxtr@warn@onexistssordo
207 \glsxtr@addloclistfield
208 \renewcommand*{\@glsxtr@autoindexcrossrefs}{}%
209 \let\@glsxtr@recordcounter\glsxtr@op@recordcounter
210 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%

```

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

```

211 \def\glsxtrsetaliasnoindex{%

```

\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.

```

212 \ifdef\gls@setupsort@none{\@gls@setupsort@none}{}%
213 }%
214 \or

```

Record and index.

```

215 \def\glsxtr@setup@record{%
216 \renewcommand*{\@do@seeglossary}{\@glsxtr@org@doseeglossary}%
217 \let\glsxtr@record\@glsxtr@record
218 \let\@do@wrglossary\glsxtr@do@wrglossary
219 \let\gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
220 \let\glsxtrundefaction\glsxtr@warn@undefaction
221 \let\glsxtr@warnonexistssordo\glsxtr@warn@onexistssordo
222 \glsxtr@addloclistfield
223 \let\@glsxtr@recordcounter\glsxtr@op@recordcounter
224 \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
225 \undef\glsxtrsetaliasnoindex
226 }%
227 \fi
228 }

```

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

```

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

```

Need to provide conditional commands that are backward compatible:

```

\if@glsxtrdocdef
230 \newcommand*{\if@glsxtrdocdef}{\ifnum\@glsxtr@docdefval>0 }

```

lsxtrdocdeftrue

```
231 \newcommand*{\@glxtrdocdeftrue}{\@glxtr@docdefval=1 }
```

lsxtrdocdeffalse

```
232 \newcommand*{\@glxtrdocdeffalse}{\@glxtr@docdefval=0 }
```

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

```
233 \define@choicekey{glossaries-extra.sty}{docdef}{\val\nr}%
234 {false,true,restricted}[true]%
235 {%
236   \@glxtr@docdefval=\nr\relax
237   \ifnum\@glxtr@docdefval=2\relax
238     \renewcommand*{\@glsdofexistsorwarn}{\glsdofexists}%
239   \fi
240 }
```

ocdefrestricted

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

oifexistsorwarn

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

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

indexcrossrefs

Automatically index cross references at the end of the document

```
243 \define@boolkey{glossaries-extra.sty}{@glxtr}{indexcrossrefs}[true]{%
244   \if@glxtrindexcrossrefs
245   \else
246     \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
247   \fi
248 }
```

Switch off since this can increase the build time.

```
249 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

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

autoseeindex

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

```
251 \define@boolkey{glossaries-extra.sty}{@glxtr@}{autoseeindex}[true]{%
252 }
253 \@glxtr@autoseeindextrue
```

iesExtraWarning Allow users to suppress warnings.

```

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

```

raWarningNoLine Allow users to suppress warnings.

```

255 \newcommand*{\GlossariesExtraWarningNoLine}[1]{%
256   \PackageWarningNoLine{glossaries-extra}{#1}}

257 \@glxtr@declareoption{nowarn}{%
258   \let\GlossariesExtraWarning\@gobble
259   \let\GlossariesExtraWarningNoLine\@gobble
260   \glxtr@doption{nowarn}%
261 }

postdot Shortcut for nopostdot=false
262 \@glxtr@declareoption{postdot}{%
263   \glxtr@doption{nopostdot=false}%
264 }

glxtrabbrvtype Glossary type for abbreviations.
265 \newcommand*{\glxtrabbrvtype}{\glsdefaulttype}

abbreviationsdef Set by abbreviations option.
266 \newcommand*{\@glxtr@abbreviationsdef}{}

abbreviationsdef
267 \newcommand*{\@glxtr@doabbreviationsdef}{%
268   \@ifpackageloaded{babel}{%
269     {\providecommand{\abbreviationsname}{\acronymname}}%
270     {\providecommand{\abbreviationsname}{Abbreviations}}%
271     \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
272     \renewcommand*{\glxtrabbrvtype}{abbreviations}%
273     \newcommand*{\printabbreviations}[1][1]{%
274       \printglossary[type=\glxtrabbrvtype,##1]%
275     }%
276     \disable@keys{glossaries-extra.sty}{abbreviations}%

    If the acronym option hasn't been used, change \acronymtype to \glxtrabbrvtype.
277     \ifglsacronym
278     \else
279       \renewcommand*{\acronymtype}{\glxtrabbrvtype}%
280     \fi
281 }%

abbreviations If abbreviations, create a new glossary type for abbreviations.
282 \@glxtr@declareoption{abbreviations}{%
283   \let\@glxtr@abbreviationsdef\@glxtr@doabbreviationsdef
284 }

```

iationShortcuts Enable shortcut commands for the abbreviations. Unlike the analogous command provided by glossaries, this uses `\newcommand` instead of `\let` as a safety feature.

```

285 \newcommand*\GlsXtrDefineAbbreviationShortcuts{%
286   \newcommand*\ab{\cglS}%
287   \newcommand*\abp{\cglSpl}%
288   \newcommand*\as{\glSxtrshort}%
289   \newcommand*\asp{\glSxtrshortpl}%
290   \newcommand*\al{\glSxtrlong}%
291   \newcommand*\alp{\glSxtrlongpl}%
292   \newcommand*\af{\glSxtrfull}%
293   \newcommand*\afp{\glSxtrfullpl}%
294   \newcommand*\Ab{\cGlS}%
295   \newcommand*\Abp{\cGlSpl}%
296   \newcommand*\As{\Glsxtrshort}%
297   \newcommand*\Asp{\Glsxtrshortpl}%
298   \newcommand*\Al{\Glsxtrlong}%
299   \newcommand*\Alp{\Glsxtrlongpl}%
300   \newcommand*\Af{\Glsxtrfull}%
301   \newcommand*\Afp{\Glsxtrfullpl}%
302   \newcommand*\AB{\cGLS}%
303   \newcommand*\ABP{\cGLSpl}%
304   \newcommand*\AS{\GLSxtrshort}%
305   \newcommand*\ASP{\GLSxtrshortpl}%
306   \newcommand*\AL{\GLSxtrlong}%
307   \newcommand*\ALP{\GLSxtrlongpl}%
308   \newcommand*\AF{\GLSxtrfull}%
309   \newcommand*\AFP{\GLSxtrfullpl}%
310   \newcommand*\newabbr{\newabbreviation}%

```

Disable this command after it's been used.

```

311 \let\GlsXtrDefineAbbreviationShortcuts\relax
312 }

```

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

```

313 \newcommand*\GlsXtrDefineOtherShortcuts{%
314   \newcommand*\newentry{\newglossaryentry}%
315   \ifdef\printsymbols
316   {%
317     \newcommand*\newsym{\glSxtrnewsymbol}%
318   }{}%
319   \ifdef\printnumbers
320   {%
321     \newcommand*\newnum{\glSxtrnewnumber}%
322   }{}%
323   \let\GlsXtrDefineOtherShortcuts\relax
324 }

```

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.

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

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

```
326 \newcommand*{\@glxtr@shortcutsval}{\ifglsacrshortcuts acro\else none\fi}%
```

Provide shortcuts option. Unlike the glossaries version, this is a choice rather than a boolean key but it also provides shortcuts=true and shortcuts=false, which are equivalent to shortcuts=all and shortcuts=none. Multiple use of this option in the *same* option list will override each other.

```
327 \define@choicekey{glossaries-extra.sty}{shortcuts}[\val\nr]%
328 {acronyms,acro,abbreviations,abbr,other,all,true,none,false}[true]{%
329   \let\@glxtr@shortcutsval\val
330   \ifcase\nr\relax % acronyms
331     \renewcommand*{\@glxtr@setupshortcuts}{%
332       \glsacrshortcutstrue
333       \DefineAcronymSynonyms
334     }%
335   \or % acro
336     \renewcommand*{\@glxtr@setupshortcuts}{%
337       \glsacrshortcutstrue
338       \DefineAcronymSynonyms
339     }%
340   \or % abbreviations
341     \renewcommand*{\@glxtr@setupshortcuts}{%
342       \GlsXtrDefineAbbreviationShortcuts
343     }%
344   \or % abbr
345     \renewcommand*{\@glxtr@setupshortcuts}{%
346       \GlsXtrDefineAbbreviationShortcuts
347     }%
348   \or % other
349     \renewcommand*{\@glxtr@setupshortcuts}{%
350       \GlsXtrDefineOtherShortcuts
351     }%
352   \or % all
353     \renewcommand*{\@glxtr@setupshortcuts}{%
354       \glsacrshortcutstrue
355       \DefineAcronymSynonyms
356       \GlsXtrDefineAbbreviationShortcuts
357       \GlsXtrDefineOtherShortcuts
358     }%
359   \or % true
360     \renewcommand*{\@glxtr@setupshortcuts}{%
361       \glsacrshortcutstrue
362       \DefineAcronymSynonyms
363       \GlsXtrDefineAbbreviationShortcuts
364       \GlsXtrDefineOtherShortcuts
365     }%
```

```

366 \else % none, false
367 \renewcommand*{\@glsxtr@setupshortcuts}{}%
368 \fi
369 }

\lsxtr@doaccsupp
370 \newcommand*{\@glsxtr@doaccsupp}{%

accsupp If accsupp, load glossaries-accsupp package.
371 \@glsxtr@declareoption{accsupp}{%
372 \renewcommand*{\@glsxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

GlossaryWarning Warning text displayed in document if the external glossary file given by the argument is miss-
ing.
373 \newcommand{\glsxtrNoGlossaryWarning}[1]{%
374 \@glsxtr@defaultnoglossarywarning{#1}%
375 }

nomissingglsstext If true, suppress the text produced if the external glossary file is missing.
376 \define@choicekey{glossaries-extra.sty}{nomissingglsstext}[\val\nr]%
377 {true,false}[true]{%
378 \ifcase\nr\relax % true
379 \renewcommand{\glsxtrNoGlossaryWarning}[1]{%
380 \null
381 }%
382 \else % false
383 \renewcommand{\glsxtrNoGlossaryWarning}[1]{%
384 \@glsxtr@defaultnoglossarywarning{#1}%
385 }%
386 \fi
387 }

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

\lsxtr@redefstyles
388 \newcommand*{\@glsxtr@redefstyles}{%

stylemods
389 \define@key{glossaries-extra.sty}{stylemods}{%
390 \ifblank{#1}%
391 {%
392 \renewcommand*{\@glsxtr@redefstyles}{%
393 \RequirePackage{glossaries-extra-stylemods}}%
394 }%
395 {%
396 \renewcommand*{\@glsxtr@redefstyles}{}%
397 \@for\@glsxtr@tmp:=#1\do{%
398 \IfFileExists{glossary-\@glsxtr@tmp.sty}%

```

```

399      {%
400        \eappto\@glxtr@redefstyles{%
401          \noexpand\RequirePackage{glossary-\@glxtr@tmp}}%
402      }%
403      {%
404        \PackageError{glossaries-extra}%
405          {Glossaries style package ‘glossary-\@glxtr@tmp.sty’
406            doesn’t exist (did you mean to use the ‘style’ key?)}%
407          {The list of values (#1) in the ‘stylemods’ key should
408            match the glossary-xxx.sty files provided with
409            glossaries.sty}%
410      }%
411    }%
412    \appto\@glxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
413  }%
414}

```

glxtr@do@style

```

415 \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.

```

416 \define@key{glossaries-extra.sty}{style}{%
417   \renewcommand*{\@glxtr@do@style}{%

```

Set this as the default style:

```

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

```

Set this style:

```

419   \setglossarystyle{#1}%
420 }%
421 }

```

Pass all other options to glossaries.

```

422 \DeclareOptionX*{%
423   \expandafter\glxtr@doooption\expandafter{\CurrentOption}}

```

Process options.

```

424 \ProcessOptionsX

```

Load glossaries if not already loaded.

```

425 \RequirePackage{glossaries}

```

Load the glossaries-accsupp package if required.

```

426 \@glxtr@doaccsupp

```

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

```

427 \let\@glxtr@org@doseeglossary\@do@seeglossary

```



@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.)

```
428 \let\@glxtr@org@gloautosee\@glo@autosee
```

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

```
429 \if@glxtr@autoseeindex
430 \else
431   \ifdef\@glxtr@org@gloautosee
432   {}%
433   {\PackageError{glossaries-extra}{‘autoseeindex=false’ package
434     option requires at least v4.30 of glossaries.sty}%
435     {You need to update the glossaries.sty package}%
436   }
437 \fi
```

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

```
438 \ifdef\@glo@autosee
439 {%
440   \renewcommand*{\@glo@autosee}{%
441     \if@glxtr@autoseeindex\@glxtr@org@gloautosee\fi}%
442 }%
443 {}
```

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

```
444 \renewcommand*{\gls@checkseeallowed}{%
445   \if@glxtr@autoseeindex\@gls@see@noindex\fi
446 }
```

Define abbreviations glossaries if required.

```
447 \@glxtr@abbreviationsdef
448 \let\@glxtr@abbreviationsdef\relax
```

Setup shortcuts if required.

```
449 \@glxtr@setupshortcuts
```

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

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

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

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

Now define the user command:

```
452 \newcommand*{\glossariesextrasetup}[1]{%
453   \let\glxtr@setup@record\relax
454   \let\@glxtr@setupshortcuts\relax
```

```

455 \let\@glxtr@redef@forglsentries\relax
456 \setkeys{glossaries-extra.sty}{#1}%
457 \@glxtr@abbreviationsdef
458 \let\@glxtr@abbreviationsdef\relax
459 \@glxtr@setupshortcuts
460 \glxtr@setup@record
461 \@glxtr@redef@forglsentries
462 }

```

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

```

463 \let\glxtr@do@wrglossary\@do@wrglossary

```

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

```

464 \let\glxtr@saveentrycounter\@gl@saveentrycounter

```

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

```

465 \let\@gl@saveentrycounter\glxtr@indexonly@saveentrycounter

```

Set up record option if required.

```

466 \glxtr@setup@record

```

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

```

467 \AtBeginDocument{%
468   \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
469   \def\@glxtrundeftag{\glxtrundeftag}%
470 }

```

## 1.2 Extra Utilities

`\glxtrifemptyglossary` `\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.)

```

471 \newcommand{\glxtrifemptyglossary}[3]{%
472   \ifcsdef{glolist@#1}%
473   {%
474     \ifcsstring{glolist@#1}{,}{#2}{#3}%
475   }%
476   {%

```

```

477     \glxtrundefaction{Glossary type ‘#1’ doesn’t exist}{}%
478     #2%
479 }%
480 }

xtrifkeydefined Tests if the key given in the first argument has been defined.
481 \newcommand*{\glxtrifkeydefined}[3]{%
482   \key@ifundefined{glossentry}{#1}{#3}{#2}%
483 }

providestoragekey Like \glxaddstoragekey but does nothing if the key has already been defined.
484 \newcommand*{\glxtrprovidestoragekey}{%
485   \@ifstar\sglsxtr@provide@storagekey\glxtr@provide@storagekey
486 }

vide@storagekey Unstarred version.
487 \newcommand*{\@glxtr@provide@storagekey}[3]{%
488   \key@ifundefined{glossentry}{#1}%
489   {%
490     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
491     \appto\@gls@keymap{,{#1}{#1}}%
492     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
493     \appto\@newglossaryentryposthook{%
494       \letcs{\@glo@tmp}{@glo@#1}%
495       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
496     }%

    Allow the user to omit the user level command if they only intended fetching the value with
    \glxtrusefield
497     \ifblank{#3}
498     {}%
499     {%
500       \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
501     }%
502   }%
503   {%

    Provide the no-link command if not already defined.
504     \ifblank{#3}
505     {}%
506     {%
507       \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
508     }%
509   }%
510 }

vide@storagekey Starred version.
511 \newcommand*{\s@glxtr@provide@storagekey}[1]{%
512   \key@ifundefined{glossentry}{#1}%

```

```

513 {%
514   \expandafter\newcommand\expandafter*\expandafter
515     {\csname gls@assign@#1@field\endcsname}[2]{%
516       \@gls@expand@field{##1}{#1}{##2}%
517     }%
518   }%
519   {}%
520   \@glsxtr@provide@addstoragekey{#1}%
521 }

```

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

`\GlsXtrFmtField`

```

522 \newcommand{\GlsXtrFmtField}{useri}

```

`\GlsXtrFmtDefaultOptions`

```

523 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

`\glsxtrfmt` The post-link hook isn't done.

```

524 \newrobustcmd*{\glsxtrfmt}[3][ ]{%
525   \glsdoifexistsordo{#2}%
526   {%
527     \ifglshasfield{\GlsXtrFmtField}{#2}%
528     {%
529       \let\do@gls@link@checkfirsthyper\relax
530       \expandafter\@gls@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
531       {\csuse{\glscurrentfieldvalue}{#3}}%
532     }%
533     {#3}%
534   }%
535   {#3}%
536 }

```

`\glsxtrentryfmt` No link or indexing.

```

537 \ifdef\texorpdfstring
538 {
539   \newcommand*{\glsxtrentryfmt}[2]{%
540     \texorpdfstring{\@glsxtrentryfmt{#1}{#2}}{#2}%
541   }
542 }
543 {
544   \newcommand*{\glsxtrentryfmt}{\@glsxtrentryfmt}
545 }

```

`@glsxtrentryfmt`

```

546 \newrobustcmd*{\@glxtrentryfmt}[2]{%
547   \glsoifexistsordo
548   {%
549     \ifglshasfield{\GlsXtrFmtField}{#1}%
550     {%
551       \csuse{\glscurrentfieldvalue}{#2}%
552     }%
553     {#2}%
554   }%
555   {#2}%
556 }

```

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

```

557 \newcommand*{\glxtrfieldlistadd}[3]{%
558   \listcsadd{glo@\glsetoklabel{#1}@#2}{#3}%
559 }

```

**trfieldlistgadd** Similarly but uses `\listcsgadd`.

```

560 \newcommand*{\glxtrfieldlistgadd}[3]{%
561   \listcsgadd{glo@\glsetoklabel{#1}@#2}{#3}%
562 }

```

**trfieldlistead** Similarly but uses `\listcseadd`.

```

563 \newcommand*{\glxtrfieldlistead}[3]{%
564   \listcseadd{glo@\glsetoklabel{#1}@#2}{#3}%
565 }

```

**trfieldlistxadd** Similarly but uses `\listcsxadd`.

```

566 \newcommand*{\glxtrfieldlistxadd}[3]{%
567   \listcsxadd{glo@\glsetoklabel{#1}@#2}{#3}%
568 }

```

Now provide commands to iterate over these lists.

**fielddolistloop**

```

569 \newcommand*{\glxtrfielddolistloop}[2]{%
570   \dolistcsloop{glo@\glsetoklabel{#1}@#2}%
571 }

```

**fieldforlistloop**

```

572 \newcommand*{\glxtrfieldforlistloop}[3]{%
573   \forlistcsloop{glo@\glsetoklabel{#1}@#2}{#3}%
574 }

```

List element tests:

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

```

575 \newcommand*{\glstrfieldifinlist}[5]{%
576   \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
577 }

```

rfieldxifinlist Expands item.

```

578 \newcommand*{\glstrfieldxifinlist}[5]{%
579   \xifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
580 }

```

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

```

581 \newcommand*{\glstrusefield}[2]{%
582   \@gls@entry@field{#1}{#2}%
583 }

```

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

```

584 \newcommand*{\Glsxtrusefield}[2]{%
585   \@gls@entry@field{#1}{#2}%
586 }

```

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

```

587 \newcommand*{\glstrdeffield}[2]{\csdef{glo@\glsdetoklabel{#1}@#2}}

```

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

```

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

```

etfieldifexists

```

589 \newcommand*{\glstrsetfieldifexists}[3]{\glsdofexists{#1}{#3}}

```

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

```

590 \newrobustcmd*{\GlsXtrSetField}[3]{%
591   \glstrsetfieldifexists{#1}{#2}%
592   {\csdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
593 }

```

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

```

594 \newrobustcmd*{\GlsXtrLetField}[3]{%
595   \glstrsetfieldifexists{#1}{#2}%
596   {\cslet{glo@\glsdetoklabel{#1}@#2}{#3}}%
597 }

```

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

```

598 \newrobustcmd*{\csGlsXtrLetField}[3]{%
599   \glstrsetfieldifexists{#1}{#2}%
600   {\csletcs{glo@\glsdetoklabel{#1}@#2}{#3}}%
601 }

```

**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.

```
602 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
603   \glstrsetfieldifexists{#1}{#2}%
604   {\csletcs{glo@\glsetoklabel{#1}@#2}{glo@\glsetoklabel{#3}@#4}}%
605 }
```

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

```
606 \newrobustcmd*{\gGlsXtrSetField}[3]{%
607   \glstrsetfieldifexists{#1}{#2}%
608   {\csgdef{glo@\glsetoklabel{#1}@#2}{#3}}%
609 }
```

**xGlsXtrSetField**

```
610 \newrobustcmd*{\xGlsXtrSetField}[3]{%
611   \glstrsetfieldifexists{#1}{#2}%
612   {\protected@csxdef{glo@\glsetoklabel{#1}@#2}{#3}}%
613 }
```

**eGlsXtrSetField**

```
614 \newrobustcmd*{\eGlsXtrSetField}[3]{%
615   \glstrsetfieldifexists{#1}{#2}%
616   {\protected@csedef{glo@\glsetoklabel{#1}@#2}{#3}}%
617 }
```

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

```
618 \ifglentrycounter
619   \newcommand*{\glstrpageref}[1]{\pageref{glentry-\glsetoklabel{#1}}}
620 \else
621   \ifglssubentrycounter
622     \newcommand*{\glstrpageref}[1]{\pageref{glentry-\glsetoklabel{#1}}}
623   \else
624     \newcommand*{\glstrpageref}[1]{\gls{#1}}
625   \fi
626 \fi
```

**lossarypreamble**

```
627 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
628   \ifcsdef{glolist@#1}%
629     {%
630       \ifcsundef{@glossarypreamble@#1}%
631       {\csdef{@glossarypreamble@#1}{}}%
632     }%
633   \csappto{@glossarypreamble@#1}{#2}%
634   }%
635   {%
636     \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
```

```

637 }%
638 }

```

lossarypreamble

```

639 \newcommand{\preglossarypreamble}[2][\glsdefaultttype]{%
640   \ifcsdef{glolist@#1}%
641   {%
642     \ifcsundef{@glossarypreamble@#1}%
643     {\csdef{@glossarypreamble@#1}{}}%
644     {}%
645     \cspreteto{@glossarypreamble@#1}{#2}%
646   }%
647   {%
648     \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
649   }%
650 }

```

### 1.3 Modifications to Commands Provided by glossaries

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

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

ewglossaryentry

```

651 \renewcommand*{\longnewglossaryentry}{%
652   \@ifstar{\glxstr@s@longnewglossaryentry\glxstr@longnewglossaryentry
653 }

```

ewglossaryentry Starred version.

```

654 \newcommand{\glxstr@s@longnewglossaryentry}[3]{%
655   \glsdoifnoexists{#1}%
656   {%
657     \bgroup
658     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
659     \long\def\@newglossaryentryprehook{%
660       \long\def\@glo@desc{#3}%
661       \@org@newglossaryentryprehook
662     }%
663     \renewcommand*{\gls@assign@desc}[1]{%
664       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
665       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
666     }
667     \gls@defglossaryentry{#1}{#2}%
668   \egroup
669 }%
670 }

```



ewglossaryentry    Unstarred version.

```
671 \newcommand{\@glxtr@longnewglossaryentry}[3]{%
672   \glsdoifnoexists{#1}%
673   {%
674     \bgroup
675     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
676     \long\def\@newglossaryentryprehook{%
677       \long\def\@glo@desc{#3\glxtrpostlongdescription}%
678       \@org@newglossaryentryprehook
679     }%
680     \renewcommand*\@gls@assign@desc[1]{%
681       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
682       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
683     }
684     \gls@defglossaryentry{#1}{#2}%
685   \egroup
686 }%
687 }
```

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

```
688 \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.

```
689 \renewcommand{\newignoredglossary}{%
690   \@ifstar\glxtr@s@newignoredglossary\glxtr@org@newignoredglossary
691 }
```

ignoredglossary    The original definition is patched to check for existence.

```
692 \newcommand*\@glxtr@org@newignoredglossary[1]{%
693   \ifcsdef{glolist@#1}
694   {%
695     \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
696   }%
697   {%
698     \ifdefempty\@ignored@glossaries
699     {%
700       \edef\@ignored@glossaries{#1}%
701     }%
702     {%
703       \eappto\@ignored@glossaries{,#1}%
704     }%
705     \csgdef{glolist@#1}{,}%
706     \ifcsundef{gls@#1@entryfmt}%
707     {%
```

```

708     \defglentryfmt[#1]{\glentryfmt}%
709 }%
710 {}%
711 \ifdefempty\@gls@nohyperlist
712 {%
713     \renewcommand*{\@gls@nohyperlist}{#1}%
714 }%
715 {%
716     \eappto\@gls@nohyperlist{, #1}%
717 }%
718 }%
719 }

```

ignoredglossary Starred form.

```

720 \newcommand*{\glstr@s@newignoredglossary}[1]{%
721     \ifcsdef{glolist@#1}
722     {%
723         \glstrundefaction{Glossary type ‘#1’ already exists}{}%
724     }%
725     {%
726         \ifdefempty\@ignored@glossaries
727         {%
728             \edef\@ignored@glossaries{#1}%
729         }%
730         {%
731             \eappto\@ignored@glossaries{, #1}%
732         }%
733         \csgdef{glolist@#1}{,}%
734         \ifcsundef{gls@#1@entryfmt}%
735         {%
736             \defglentryfmt[#1]{\glentryfmt}%
737         }%
738         {}%
739     }%
740 }

```

\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.

```

741 \glusifusetranslator
742 {%
743     \renewcommand*{\glsettoctitle}[1]{%
744         \ifcsdef{gls@tr@set@#1@toctitle}%
745         {%
746             \csuse{gls@tr@set@#1@toctitle}%
747         }%
748         {%
749             \ifcsdef{@glotype@#1@title}%
750             {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
751             {\def\glossarytoctitle{\glossarytitle}}%

```

```

752 }%
753 }%
754 }
755 {
756 \renewcommand*{\glsettoctitle}[1]{%
757 \ifcsdef{@glotype@#1@title}%
758 {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
759 {\def\glossarytoctitle{\glossarytitle}}%
760 }
761 }

```

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

```

762 \newcommand{\provideignoredglossary}{%
763 \@ifstar\glxtr@s@provideignoredglossary\glxtr@provideignoredglossary
764 }

```

ignoredglossary Unstarred version.

```

765 \newcommand*{\glxtr@provideignoredglossary}[1]{%
766 \ifcsdef{glolist@#1}
767 {}%
768 {%
769 \ifdefempty\@ignored@glossaries
770 {%
771 \edef\@ignored@glossaries{#1}%
772 }%
773 {%
774 \eappto\@ignored@glossaries{,#1}%
775 }%
776 \csgdef{glolist@#1}{,}%
777 \ifcsundef{gls@#1@entryfmt}%
778 {%
779 \defglsentryfmt[#1]{\glsentryfmt}%
780 }%
781 {}%
782 \ifdefempty\@gls@nohyperlist
783 {%
784 \renewcommand*{\@gls@nohyperlist}{#1}%
785 }%
786 {%
787 \eappto\@gls@nohyperlist{,#1}%
788 }%
789 }%
790 }

```

ignoredglossary Starred form.

```

791 \newcommand*{\glxtr@s@provideignoredglossary}[1]{%
792 \ifcsdef{glolist@#1}
793 {}%
794 {%

```

```

795 \ifdefempty\@ignored@glossaries
796 {%
797 \edef\@ignored@glossaries{#1}%
798 }%
799 {%
800 \eappto\@ignored@glossaries{,#1}%
801 }%
802 \csgdef{glolist@#1}{,}%
803 \ifcsundef{gls@#1@entryfmt}%
804 {%
805 \defglsentryfmt[#1]{\glsentryfmt}%
806 }%
807 {}%
808 }%
809 }

```

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

```

810 \newcommand*\xtrcopytoglossary[2]{%
811 \glsdoifexists{#1}%
812 {%
813 \ifcsdef{glolist@#2}
814 {%
815 \cseappto{glolist@#2}{#1,}%
816 }%
817 {%
818 \glsxtrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
819 }%
820 }%
821 }

```

### 1.3.1 Existence Checks

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

```

822 \renewcommand{\glsdoifexists}[2]{%
823 \ifglsentryexists{#1}{#2}%
824 {%

```

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

```

825 \edef\glslabel{\glsdetoklabel{#1}}%
826 \glsxtrundefaction{Glossary entry ‘\glslabel’
827 has not been defined}{You need to define a glossary entry before
828 you can reference it.}%
829 }%
830 }

```

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

```

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

```

```

832 \ifglstryexists{#1}{%
833   \glstrundefaction{Glossary entry '\glstetoklabel{#1}'
834     has already been defined}{}}{#2}%
835 }

```

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

```

836 \ifdef\glstdoifexistsordo
837 {%
838   \renewcommand{\glstdoifexistsordo}[3]{%
839     \ifglstryexists{#1}{#2}%
840     {%
841       \glstrundefaction{Glossary entry '\glstetoklabel{#1}'
842         has not been defined}{You need to define a glossary entry
843         before you can use it.}%
844       #3%
845     }%
846   }%
847 }
848 {%
849   \glstr@warnonexistsordo\glstdoifexistsordo
850   \newcommand{\glstdoifexistsordo}[3]{%
851     \ifglstryexists{#1}{#2}%
852     {%
853       \glstrundefaction{Glossary entry '\glstetoklabel{#1}'
854         has not been defined}{You need to define a glossary entry
855         before you can use it.}%
856       #3%
857     }%
858   }%
859 }

```

`\arynoexistsordo` Similarly for `\doifglossarynoexistsordo`.

```

860 \ifdef\doifglossarynoexistsordo
861 {%
862   \renewcommand{\doifglossarynoexistsordo}[3]{%
863     \ifglossaryexists{#1}%
864     {%
865       \glstrundefaction{Glossary type '#1' already exists}{}%
866       #3%
867     }%
868     {#2}%
869   }%
870 }
871 {%
872   \glstr@warnonexistsordo\doifglossarynoexistsordo
873   \newcommand{\doifglossarynoexistsordo}[3]{%
874     \ifglossaryexists{#1}%
875     {%

```

```

876     \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
877     #3%
878     }%
879     {#2}%
880     }%
881 }
882

```

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.

```

883 \appto\@newglossaryentryposthook{%
884   \ifdefvoid\@glo@see
885     {\csxdef{glo@\@glo@label @see}{}}%
886     {%
887       \csxdef{glo@\@glo@label @see}{\@glo@see}%
888       \if@glxtr@autoseeindex
889         \@glxtr@autoindexcrossrefs
890       \fi
891     }%
892 }
893 \appto\@gls@keymap{,{see}{see}}

```

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

```

894 \newcommand*{\glxtrusesee}[1]{%
895   \glsdoifexists{#1}%
896   {%
897     \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
898     \ifdefempty\@glo@see
899       {}%
900     {%
901       \expandafter\glxtr@usesee\@glo@see\@end@glxtr@usesee
902     }%
903   }%
904 }

```

\glxtr@usesee

```

905 \newcommand*{\glxtr@usesee}[1][\seename]{%
906   \@glxtr@usesee{#1}%
907 }

```

\@glxtr@usesee

```

908 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
909   \glxtruseseeformat{#1}{#2}%
910 }

```

`\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.

```
911 \newcommand*{\glxtruseseeformat}[2]{%
912   \glssseeformat[#1]{#2}{}%
913 }
```

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

```
914 \newcommand*{\glxtruseseealso}[1]{%
915   \glsdexists{#1}%
916   {%
917     \letcs{\@glo@see}{glo\glsdetoklabel{#1}@seealso}%
918     \ifdefempty\@glo@see
919     {%
920       {%
921         \expandafter\glxtruseseealsoformat\expandafter{\@glo@see}%
922       }%
923     }%
924 }
```

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

```
925 \newcommand*{\glssseealsoformat}[1]{%
926   \glssseeformat[\seealso]{#1}{}%
927 }
```

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

```
928 \newrobustcmd{\glxtrseeelist}[1]{%
929   \edef\@glo@tmp{\noexpand\glssseeelist{#1}}\@glo@tmp
930 }
```

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

```
931 \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.

```
932 \ifdef\@xdycrossrefhook
933 {
```

Add the cross-reference class definition to the hook.

```
934   \appto\@xdycrossrefhook{%
935     \write\glswrite{(define-crossref-class \string"seealso\string"
936       :unverified )}%
937     \write\glswrite{(markup-crossref-list
938       :class \string"seealso\string"^^J\space\space\space
939       :open \string"\string\glxtruseseealsoformat\glsoopenbrace\string"
```

```

940         :close \string\glsclosebrace\string"))}%
941     }

```

Append to class list.

```

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

```

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

```

943 \newrobustcmd*{\glxtrindexseealso}[2]{%
944   \def\@gls@xref{#2}%
945   \@onelevel@sanitize\@gls@xref
946   \@gls@checkmkidxchars\@gls@xref
947   \gls@glossary{\csname glo@#1@type\endcsname}{%
948     (indexentry
949       :tkey (\csname glo@#1@index\endcsname)
950       :xref (\string"\@gls@xref\string")
951       :attr \string"seealso\string"
952     )
953   }%
954 }
955 }
956 {

```

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

```

957 \newrobustcmd*{\glxtrindexseealso}{\glssee[\seealsoname]}
958 }

```

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`.

```

959 \ifdef\gls@set@xr@key
960 {

```

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

```

961 \define@key{glossentry}{alias}{%
962   \gls@set@xr@key{alias}{\@glo@alias}{#1}%
963 }
964 \define@key{glossentry}{seealso}{%
965   \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
966 }

```

Add to the key mappings.

```

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

```

Set the default value.

```

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

```

Assign the field values.

```

969 \appto\@newglossaryentryposthook{%

```



```

970 \ifdefvoid\@glo@seealso
971 {\csxdef{glo@\@glo@label @seealso}{}}%
972 {%
973 \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
974 \if@glxtr@autoseeindex
975 \@glxtr@autoindexcrossrefs
976 \fi
977 }%

```

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

```

978 \ifdefvoid\@glo@alias
979 {\csxdef{glo@\@glo@label @alias}{}}%
980 {%
981 \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
982 }%
983 }

```

Provide user-level commands to access the values.

`\glxtralias`

```

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

```

`trseealsolabels`

```

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

```

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

```

986 \appto\@glo@autoseehook{%
987 \ifdefvoid\@glo@alias
988 {%
989 \ifdefvoid\@glo@seealso
990 {}%
991 {%
992 \edef\@do@glsssee{\noexpand\glxtrindexseealso
993 {\@glo@label}{\@glo@seealso}}%
994 \@do@glsssee
995 }%
996 }%
997 {%

```

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

```

998 \ifdefvoid\@glo@see
999 {%
1000 \edef\@do@glsssee{\noexpand\glsssee{\@glo@label}{\@glo@alias}}%
1001 \@do@glsssee
1002 }%
1003 {}%
1004 }%
1005 }%
1006 }
1007 {

```

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

`\glsxtralias`

```
1008 \glsaddstoragekey*{alias}{\glsxtralias}
```

`trseealsolabels`

```
1009 \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.

```
1010 \appto\@newglossaryentryposthook{%
1011   \ifcsvoid{glo@\@glo@label @alias}%
1012   {%
1013     \ifcsvoid{glo@\@glo@label @seealso}%
1014     {}%
1015     {%
1016       \edef\@do@glsssee{\noexpand\glsxtrindexseealso
1017         {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1018       \@do@glsssee
1019     }%
1020   }%
1021   {%
```

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

```
1022   \ifdefvoid\@glo@see
1023   {%
1024     \edef\@do@glsssee{\noexpand\glsssee
1025       {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1026     \@do@glsssee
1027   }%
1028   {}%
1029   }%
1030 }
1031 }
```

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

```
1032 \AtEndDocument{\if@glsxtrindexcrossrefs\glsxtraddallcrossrefs\fi}
```

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

```
1033 \newcommand*\glsxtraddallcrossrefs{%
1034   \forallglossaries{\@glo@type}%
1035   {%
1036     \forglsentries[\@glo@type]{\@glo@label}%
1037     {%
1038       \ifglssused{\@glo@label}%
```

```

1039      {\expandafter\@glxtr@addunusedxrefs\expandafter{\@glo@label}}{}%
1040    }%
1041  }%
1042 }

```

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

```

1043 \newcommand*{\@glxtr@addunusedxrefs}[1]{%
1044   \letcs{\@glo@see}{glo\@glsetoklabel{#1}@see}%
1045   \ifdefvoid\@glo@see
1046     {}%
1047     {%
1048       \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1049     }%
1050   \letcs{\@glo@see}{glo\@glsetoklabel{#1}@seealso}%
1051   \ifdefvoid\@glo@see
1052     {}%
1053     {%
1054       \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
1055     }%
1056 }

```

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

```

1057 \newcommand*{\glxtr@addunused}[1][]{%
1058   \@glxtr@addunused
1059 }

```

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

```

1060 \def\@glxtr@addunused#1\@end@glxtr@addunused{%
1061   \@for\@glxtr@label:=#1\do
1062   {%
1063     \ifglxused{\@glxtr@label}{}%
1064     {%
1065       \glxadd[format=glxtrunusedformat]{\@glxtr@label}%
1066       \glxunset{\@glxtr@label}%
1067       \expandafter\@glxtr@addunusedxrefs\expandafter{\@glxtr@label}%
1068     }%
1069   }%
1070 }

```

**lxtrunusedformat**

```

1071 \newcommand*{\glxtrunusedformat}[1]{\unskip}

```

### 1.3.2 Document Definitions

**noidxglossaries** Modify \makenoidxglossaries so that it automatically switches off (unless the restricted setting is on) and disables the docdef key.

```

1072 \let\glxtr@orgmakenoidxglossaries\makenoidxglossaries

```

```

1073 \renewcommand{\makenoidxglossaries}{%
1074   \glstr@orgmakenoidxglossaries
1075   \if@glstrdocdefrestricted

```

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

```

1076   \renewcommand*{\@gls@reference}[3]{%
1077     \ifcsundef{@glsref@##1}{\csgdef{@glsref@##1}{}}{}%
1078     \ifinlistcs{##2}{@glsref@##1}%
1079     {}%
1080     {\listcsgadd{@glsref@##1}{##2}}%
1081     \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
1082     {\csgdef{glo@\glsdetoklabel{##2}@loclist}{}}%
1083     {}%
1084     \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
1085   }%
1086 \else

```

Disable document definitions.

```

1087   \@glstrdocdeffalse
1088 \fi
1089 \disable@keys{glossaries-extra.sty}{docdef}%
1090 }

```

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

```

1091 \renewcommand*{\@gls@defdocnewglossaryentry}{%
1092   \ifcase \@glstr@docdefval
1093   docdef=false:
1094   \renewcommand*{\newglossaryentry}[2]{%
1095     \PackageError{glossaries-extra}{Glossary entries must
1096       be \MessageBreak defined in the preamble with \MessageBreak
1097       package option 'docdef=false'\MessageBreak(consider using
1098       'docdef=restricted')}{Move your glossary definitions to
1099       the preamble. You can also put them in a \MessageBreak separate file
1100       and load them with \string\loadglsentries.}%
1101   }%
1102 \or

```

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

```

1102   \let\gls@checkseeallowed\relax
1103   \let\newglossaryentry\new@glossaryentry
1104 \or

```

Restricted mode just needs to allow the see value.

```

1105   \let\gls@checkseeallowed\relax
1106 \fi
1107 }%

```

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`

```
1108 \newcommand*{\GlsXtrEnableOnTheFly}{%
1109   \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
1110 }
```

`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.)

```
1111 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
1112   \renewcommand*{\glsdetoklabel}[1]{%
1113     \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
1114     {%
1115       \expandafter\detokenize\expandafter{##1}%
1116     }%
1117     {\detokenize{##1}}%
1118   }%
1119   \@GlsXtrEnableOnTheFly
1120 }
1121 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
1122   \expandafter\if\glsbackslash#1%
1123     #3%
1124   \else
1125     #4%
1126   \fi
1127 }
```

`sxtrstarflywarn`

```
1128 \newcommand*{\glsxtrstarflywarn}{%
1129   \GlossariesExtraWarning{Experimental starred version of
1130   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
1131   read the warnings in the glossaries-extra user manual)}}%
1132 }
```

`rEnableOnTheFly`

```
1133 \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`

```
1134 \newcommand*{\glsxtrcat}{general}
```

```

\glsxtr
1135 \newcommand*\glsxtr}[1] [] {%
1136 \def\glsxtr@keylist{##1}%
1137 \@glsxtr
1138 }

\@glsxtr
1139 \newcommand*\@glsxtr}[2] [] {%
1140 \ifglsentryexists{##2}%
1141 {%
1142 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1143 }%
1144 {%
1145 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1146 description={\nopostdesc},##1}%
1147 }%
1148 \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
1149 }

\Glsxtr
1150 \newcommand*\Glsxtr}[1] [] {%
1151 \def\glsxtr@keylist{##1}%
1152 \@Glsxtr
1153 }

\@Glsxtr
1154 \newcommand*\@Glsxtr}[2] [] {%
1155 \ifglsentryexists{##2}%
1156 {%
1157 \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1158 }%
1159 {%
1160 \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1161 description={\nopostdesc},##1}%
1162 }%
1163 \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
1164 }

\glsxtrpl
1165 \newcommand*\glsxtrpl}[1] [] {%
1166 \def\glsxtr@keylist{##1}%
1167 \@glsxtrpl
1168 }

\@glsxtrpl
1169 \newcommand*\@glsxtrpl}[2] [] {%
1170 \ifglsentryexists{##2}%
1171 {%

```

```

1172     \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1173 }%
1174 {%
1175     \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1176     description={\nopostdesc},##1}%
1177 }%
1178 \expandafter\glsp\expandafter[\glstr@keylist]{##2}%
1179 }

```

\Glsxtrpl

```

1180 \newcommand*\Glsxtrpl[1][]{%
1181     \def\glstr@keylist{##1}%
1182     \@Glsxtrpl
1183 }

```

\@Glsxtrpl

```

1184 \newcommand*\@Glsxtrpl[2][]{%
1185     \ifglstryexists{##2}
1186     {%
1187         \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1188     }%
1189     {%
1190         \gls@defglossaryentry{##2}{name={##2},category=\glstrcat,
1191         description={\nopostdesc},##1}%
1192     }%
1193     \expandafter\glsp\expandafter[\glstr@keylist]{##2}%
1194 }

```

\GlsXtrWarning

```

1195 \newcommand*\GlsXtrWarning[2]{%
1196     \def\glstr@optlist{##1}%
1197     \@onelevel@sanitize\glstr@optlist
1198     \GlossariesExtraWarning{The options '\glstr@optlist' have
1199     been ignored for entry '##2' as it has already been defined}%
1200 }

```

Disable commands after the glossary:

```

1201 \renewcommand\@printglossary[2]{%
1202     \def\glstr@printglossopts{##1}%
1203     \@glstr@orgprintglossary{##1}{##2}%
1204     \def\glstr{\glstr@disabledflycommand\glstr}%
1205     \def\glstrpl{\glstr@disabledflycommand\glstrpl}%
1206     \def\Glsxtr{\glstr@disabledflycommand\Glsxtr}%
1207     \def\Glsxtrpl{\glstr@disabledflycommand\Glsxtrpl}%
1208 }

```

\glstr@disabledflycommand

```

1209 \newcommand*\glstr@disabledflycommand[1]{%
1210     \PackageError{glossaries-extra}%

```

```

1211 {\string##1\space can't be used after any of the \MessageBreak
1212 glossaries have been displayed}%
1213 {The on-the-fly commands enabled by
1214 \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1215 before the glossaries. If you want to use any entries \MessageBreak
1216 after any of the glossaries, you must use the standard \MessageBreak
1217 method of first defining the entry and then using the \MessageBreak
1218 entry with commands like \string\gls}%
1219 @@glxtr@disabledflycommand
1220 }%
1221 \newcommand*{@@glxtr@disabledflycommand}[2][\##2]

```

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

```

1222 \let\GlsXtrEnableOnTheFly\relax
1223 }
1224 \@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.

```

1225 \newcommand*{\@glxtr@current@style}{\@glossary@default@style}

```

Modify \setglossarystyle to set the above.

etglossarystyle

```

1226 \renewcommand*{\setglossarystyle}[1]{%
1227 \ifcsundef{@glsstyle@#1}%
1228 {%
1229 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
1230 }%
1231 {%
1232 \csname @glsstyle@#1\endcsname

```

Only set the current style if it exists.

```

1233 \protected@edef\@glxtr@current@style{#1}%
1234 }%
1235 \ifx\@glossary@default@style\relax
1236 \protected@edef\@glossary@default@style{#1}%
1237 \fi
1238 }

```

In case we have an old version of glossaries:

```

1239 \ifdef\@glossary@default@style
1240 {}
1241 {%
1242 \let\@glossary@default@style\relax
1243 }

```



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

```

1244 \ifdef\glslistdottedwidth
1245 {%
1246   \ifdim\glslistdottedwidth=.5\hsize
1247     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
1248   \AtBeginDocument{%
1249     \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
1250       \setlength{\glslistdottedwidth}{.5\columnwidth}%
1251     \fi
1252   }%
1253 \fi
1254 }
1255 {}%
```

Similarly for \glsdescwidth:

\glsdescwidth

```

1256 \ifdef\glsdescwidth
1257 {%
1258   \ifdim\glsdescwidth=.6\hsize
1259     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
1260   \AtBeginDocument{%
1261     \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
1262       \setlength{\glsdescwidth}{.6\columnwidth}%
1263     \fi
1264   }%
1265 \fi
1266 }
1267 {}%
```

and for \glspagelistwidth:

lspagelistwidth

```

1268 \ifdef\glspagelistwidth
1269 {%
1270   \ifdim\glspagelistwidth=.1\hsize
1271     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
1272   \AtBeginDocument{%
1273     \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
1274       \setlength{\glspagelistwidth}{.1\columnwidth}%
1275     \fi
1276   }%
1277 \fi
1278 }
1279 {}%
```

aryentrynumbers Has the nonumberlist option been used?

```

1280 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
```

```

1281 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
1282   \glsnonumberlistfalse
1283   \renewcommand*{\glossaryentrynumbers}[1]{%
1284     \ifglsentryexists{\glscurrententrylabel}%
1285     {%
1286       \@glsxtrpreloctag
1287       \GlsXtrFormatLocationList{#1}%
1288       \@glsxtrpostloctag
1289       \gls@save@numberlist{#1}%
1290     }{}%
1291   }%
1292 \else
1293   \glsnonumberlisttrue
1294   \renewcommand*{\glossaryentrynumbers}[1]{%
1295     \ifglsentryexists{\glscurrententrylabel}%
1296     {%
1297       \gls@save@numberlist{#1}%
1298     }{}%
1299   }%
1300 \fi

```

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

```

1301 \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**

```

1302 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
1303   \let\@glsxtrpreloctag\@glsxtrpreloctag
1304   \let\@glsxtrpostloctag\@glsxtrpostloctag
1305   \renewcommand*{\@glsxtr@pagetag}{#1}%
1306   \renewcommand*{\@glsxtr@pagetag}{#2}%
1307   \renewcommand*{\@glsxtr@savepreloctag}[2]{%
1308     \csgdef{\@glsxtr@preloctag@##1}{##2}%
1309   }%
1310   \renewcommand*{\@glsxtr@doloctag}{%
1311     \ifcsundef{\@glsxtr@preloctag@\glscurrententrylabel}%
1312     {%
1313       \GlossariesWarning{Missing pre-location tag for ‘\glscurrententrylabel’.
1314         Rerun required}%
1315     }%
1316     {%
1317       \csuse{\@glsxtr@preloctag@\glscurrententrylabel}%
1318     }%

```

```

1319 }%
1320 }
1321 \@onlypreamble\GlsXtrEnablePreLocationTag

```

glsxtrpreloctag

```

1322 \newcommand*{\@glsxtrpreloctag}{%
1323   \let\@glsxtr@org@delimN\delimN
1324   \let\@glsxtr@org@delimR\delimR
1325   \let\@glsxtr@org@glsignore\glsignore
   \gdef is required as the delimiters may occur inside a scope.
1326   \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1327   \renewcommand*{\delimN}{%
1328     \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1329     \@glsxtr@org@delimN}%
1330   \renewcommand*{\delimR}{%
1331     \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
1332     \@glsxtr@org@delimR}%
1333   \renewcommand*{\glsignore}[1]{%
1334     \gdef\@glsxtr@thisloctag{\relax}%
1335     \@glsxtr@org@glsignore{##1}}%
1336   \@glsxtr@doloctag
1337 }

```

glsxtrpreloctag

```

1338 \newcommand*{\@glsxtrpreloctag}{}

```

@glsxtr@pagetag

```

1339 \newcommand*{\@glsxtr@pagetag}{}%

```

glsxtr@pagetag

```

1340 \newcommand*{\@glsxtr@pagetag}{}%

```

lsxtrpostloctag

```

1341 \newcommand*{\@glsxtrpostloctag}{%
1342   \let\delimN\@glsxtr@org@delimN
1343   \let\delimR\@glsxtr@org@delimR
1344   \let\glsignore\@glsxtr@org@glsignore
1345   \protected@write\@auxout{%
1346     {\string\@glsxtr@savepreloctag{\glscurrententrylabel}\@glsxtr@thisloctag}}%
1347 }

```

lsxtrpostloctag

```

1348 \newcommand*{\@glsxtrpostloctag}{}

```

lsxtr@preloctag

```

1349 \newcommand*{\@glsxtr@savepreloctag}[2]{}
1350 \protected@write\@auxout{%
1351   \string\providecommand\string\@glsxtr@savepreloctag[2]{}

```

glsxtr@doloctag

```
1352 \newcommand*{\@glsxtr@doloctag}{}
```

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

```
1353 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
1354   \XKV@plfalse
1355   \XKV@sttrue
1356   \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
1357   {%
1358     \csname glsnonumberlist\XKV@resa\endcsname
1359     \ifglsnonumberlist
1360       \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
1361     \else
1362       \def\glossaryentrynumbers##1{%
1363         \@glsxtrpreloctag
1364         \GlsXtrFormatLocationList{##1}%
1365         \@glsxtrpostloctag
1366         \gls@save@numberlist{##1}}%
1367     \fi
1368   }%
1369 }
```

### 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.

```
1370 \renewcommand*{\glsentryfmt}{%
1371   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}{}}%
1372   \glsifregular{\glslabel}%
1373   {\glsxtrregularfont{\glsgenentryfmt}}%
1374   {%
1375     \ifglshasshort{\glslabel}%
1376     {\glsxtrgenabbrvfmt}%
1377     {\glsxtrregularfont{\glsgenentryfmt}}%
1378   }%
1379 }
```

sxtrregularfont Font used for regular entries.

```
1380 \newcommand*{\glsxtrregularfont}[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 `\glsxtrifwasfirstuse` etc to allow the postlink hook to work better. This now has an optional argument that sets up the defaults.

```
1381 \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 enter exists.

```
1382 \@glsxtr@record{#2}{#3}{glslink}%
1383 \glsdoifexists{#3}%
1384 {%
```

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).

```
1385 \let\glsxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
1386 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1387 \def\glscustomtext{#4}%
1388 \@glsxtr@field@linkdefs
1389 #1%
1390 \@gls@link[#2]{#3}{#4}%
1391 \let\ifKV@glslink@hyper\glsxtrorg@ifKV@glslink@hyper
1392 }%
1393 \glspostlinkhook
1394 }
```

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.

```
1395 \let\@glsxtr@org@gls@\@gls@
1396 \def\@gls@#1#2{%
1397 \@glsxtr@record{#1}{#2}{glslink}%
1398 \@glsxtr@org@gls@{#1}{#2}%
1399 }%
```

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

```
1400 \let\@glsxtr@org@glspl@\@glspl@
1401 \def\@glspl@#1#2{%
1402 \@glsxtr@record{#1}{#2}{glslink}%
1403 \@glsxtr@org@glspl@{#1}{#2}%
1404 }%
```

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

```
1405 \let\@glsxtr@org@Gls@\@Gls@
1406 \def\@Gls@#1#2{%
1407 \@glsxtr@record{#1}{#2}{glslink}%
1408 \@glsxtr@org@Gls@{#1}{#2}%
1409 }%
```

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

```
1410 \let\@glxtr@org@Glspl@\@Glspl@
1411 \def\@Glspl@#1#2{%
1412   \@glxtr@record{#1}{#2}{glslink}%
1413   \@glxtr@org@Glspl@{#1}{#2}%
1414 }%
```

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

```
1415 \let\@glxtr@org@GLS@\@GLS@
1416 \def\@GLS@#1#2{%
1417   \@glxtr@record{#1}{#2}{glslink}%
1418   \@glxtr@org@GLS@{#1}{#2}%
1419 }%
```

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

```
1420 \let\@glxtr@org@GLSpl@\@GLSpl@
1421 \def\@GLSpl@#1#2{%
1422   \@glxtr@record{#1}{#2}{glslink}%
1423   \@glxtr@org@GLSpl@{#1}{#2}%
1424 }%
```

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

```
1425 \renewcommand*{\@glsdisp}[3][{}]{%
1426   \@glxtr@record{#1}{#2}{glslink}%
1427   \glsdoifexists{#2}{%
1428     \let\do@gl@link@checkfirsthyper\@gl@link@checkfirsthyper
1429     \let\glsifplural\@secondoftwo
1430     \let\glscapscase\@firstofthree
1431     \def\glscustomtext{#3}%
1432     \def\glsinsert{}%
1433     \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
1434     \@gl@link[#1]{#2}{\@glo@text}%
1435     \ifKV@glslink@local
1436       \glslocalunset{#2}%
1437     \else
1438       \glsunset{#2}%
1439     \fi
1440   }%
1441   \glspostlinkhook
1442 }
```

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

```
1443 \renewcommand*{\@gls@link}[3][{}]{%
1444   \@glxtr@record{#1}{#2}{glslink}%
1445   \glsdoifexistsordo{#2}%
1446   {%
1447     \let\do@gl@link@checkfirsthyper\relax
```

```

1448 \gls@link[#1]{#2}{#3}%
1449 }%
1450 {%
1451 \glstextformat{#3}%
1452 }%
1453 \glspostlinkhook
1454 }

```

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

```

1455 \newcommand*{\glsxtrinitwrgloss}{%
1456 \glsifattribute{\glslabel}{wrgloss}{after}%
1457 {%
1458 \glsxtrinitwrglossbeforefalse
1459 }%
1460 {%
1461 \glsxtrinitwrglossbeforetrue
1462 }%
1463 }

```

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

```

1464 \newif\ifglsxtrinitwrglossbefore
1465 \glsxtrinitwrglossbeforetrue

```

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

```

1466 \define@choicekey{glslink}{wrgloss}[\val\nr]{before,after}%
1467 {%
1468 \ifcase\nr\relax
1469 \glsxtrinitwrglossbeforetrue
1470 \or
1471 \glsxtrinitwrglossbeforefalse
1472 \fi
1473 }

```

**\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.

```

1474 \def\gls@link[#1]#2#3{%
1475 \leavevmode
1476 \edef\glslabel{\glsdetoklabel{#2}}%
1477 \def\gls@link@opts{#1}%
1478 \let\gls@link@label\glslabel
1479 \def\glsnumberformat{glsnumberformat}%
1480 \edef\gls@counter{\csname glo@\glslabel @counter\endcsname}%
1481 \edef\glstype{\csname glo@\glslabel @type\endcsname}%
1482 \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper

```

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

```

1483 \glsxtrinitwrgloss

```

As the original definition. Note that the default link options may override \glxtrinitwrgloss.

```
1484 \@gls@setdefault@glslink@opts
1485 \do@gl:disablehyperinlist
1486 \do@gl@link@checkfirsthyper
1487 \setkeys{glslink}{#1}%
1488 \glslinkpostsetkeys
1489 \@gls@saveentrycounter
1490 \@gls@setsort{glslabel}%
```

Do write if it should occur before the link text:

```
1491 \ifglxtrinitwrglossbefore
1492 \do@wrglossary{#2}%
1493 \fi
```

Do the link text:

```
1494 \ifKV@glslink@hyper
1495 \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
1496 \else
1497 \glndonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
1498 \fi
```

Do write if it should occur after the link text:

```
1499 \ifglxtrinitwrglossbefore
1500 \else
1501 \do@wrglossary{#2}%
1502 \fi
```

As the original definition:

```
1503 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
1504 }
```

```
1505 \define@key{glossadd}{thevalue}{\def\@glxtr@thevalue{#1}}
```

```
1506 \define@key{glossadd}{theHvalue}{\def\@glxtr@theHvalue{#1}}
```

\glsadd   Redefine to include \@glxtr@record

```
1507 \renewrobustcmd*{\glsadd}[2][ ]{%
1508 \@gls@adjustmode
1509 \@glxtr@record{#1}{#2}{glossadd}%
1510 \glsdoifexists{#2}%
1511 {%
1512 \def\@glsnumberformat{glsnumberformat}%
1513 \edef\@gls@counter{\csname glo@\glsetoklabel{#2}@counter\endcsname}%
1514 \def\@glxtr@thevalue{}%
1515 \def\@glxtr@theHvalue{\@glxtr@thevalue}%
1516 \setkeys{glossadd}{#1}%
1517 \ifdefempty{\@glxtr@thevalue}%
1518 {%
1519 \@gls@saveentrycounter
1520 }%

```



```

1521   {%
1522       \let\theglstrycounter\@glstr@thevalue
1523       \def\theHglstrycounter{\@glstr@theHvalue}%
1524   }%
1525   \@@do@wrglossary{#2}%
1526 }%
1527 }

```

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

```

1528 \newcommand*{\@glstr@field@linkdefs}{%
1529   \let\glstrifwasfirstuse\@secondoftwo
1530   \let\glstrifplural\@secondoftwo
1531   \let\glscapscase\@firstofthree
1532   \let\glinsert\@empty
1533 }

```

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

**assignfieldfont**

```

1534 \newcommand*{\glstrassignfieldfont}[1]{%
1535   \ifglstryexists{#1}%
1536   {%
1537     \ifglshasshort{#1}%
1538     {%
1539       \glsetabbrvfmt{\glscategory{#1}}%
1540       \glstrifregular{#1}%
1541       {\let\@gls@field@font\glstrregularfont}%
1542       {\let\@gls@field@font\@firstofone}%
1543     }%
1544     {%
1545       \glstrifnotregular{#1}%
1546       {\let\@gls@field@font\@firstofone}%
1547       {\let\@gls@field@font\glstrregularfont}%
1548     }%
1549   }%
1550   {%
1551     \let\@gls@field@font\@gobble
1552   }%
1553 }

```

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

```

1554 \def\@glstext@#1#2[#3]{%
1555   \glstrassignfieldfont{#2}%
1556   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
1557 }

```

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

```

1558 \def\@GLStext@#1#2[#3]{%

```

```

1559 \glstrassignfieldfont{#2}%
1560 \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
1561 {\@gls@field@font{\Glsaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
1562 }

```

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

```

1563 \def\@Glstext@#1#2[#3]{%
1564 \glstrassignfieldfont{#2}%
1565 \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
1566 {\@gls@field@font{\Glsaccesstext{#2}#3}}%
1567 }

```

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`

```

1568 \newcommand*{\glstrchecknohyperfirst}[1]{%
1569 \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
1570 }

```

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

```

1571 \def\@glsfirst@#1#2[#3]{%
1572 \glstrassignfieldfont{#2}%
1573 \gls@field@link
1574 [\let\glstrifwasfirstuse\@firstoftwo
1575 \glstrchecknohyperfirst{#2}%
1576 ]{#1}{#2}%
1577 {\@gls@field@font{\glsaccessfirst{#2}#3}}%
1578 }

```

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

```

1579 \def\@Glsfirst@#1#2[#3]{%
1580 \glstrassignfieldfont{#2}%
1581 \gls@field@link
1582 [\let\glstrifwasfirstuse\@firstoftwo
1583 \let\glscapscase\@secondofthree
1584 \glstrchecknohyperfirst{#2}%
1585 ]%
1586 {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
1587 }

```

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

```

1588 \def\@GLSfirst@#1#2[#3]{%
1589 \glstrassignfieldfont{#2}%

```

Ensure that \GLSfirst honours the nohyperfirst attribute.

```
1590 \@gls@field@link
1591 [\let\glstrifwasfirstuse\@firstoftwo
1592 \let\glscapscase\@thirdofthree
1593 \glstrchecknohyperfirst{#2}%
1594 ]%
1595 {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
1596 }
```

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

```
1597 \def\@glsplural@#1#2[#3]{%
1598 \glstrassignfieldfont{#2}%
1599 \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
1600 {\@gls@field@font{\glsaccessplural{#2}#3}}%
1601 }
```

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

```
1602 \def\@Glsplural@#1#2[#3]{%
1603 \glstrassignfieldfont{#2}%
1604 \@gls@field@link
1605 [\let\glsifplural\@firstoftwo
1606 \let\glscapscase\@secondofthree
1607 ]%
1608 {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
1609 }
```

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

```
1610 \def\@GLSplural@#1#2[#3]{%
1611 \glstrassignfieldfont{#2}%
1612 \@gls@field@link
1613 [\let\glsifplural\@firstoftwo
1614 \let\glscapscase\@thirdofthree
1615 ]%
1616 {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
1617 }
```

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

```
1618 \def\glsfirstplural@#1#2[#3]{%
1619 \glstrassignfieldfont{#2}%
1620 \@gls@field@link
1621 [\let\glstrifwasfirstuse\@firstoftwo
1622 \let\glsifplural\@firstoftwo
1623 \glstrchecknohyperfirst{#2}%
1624 ]%
1625 {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
1626 }
```

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

```
1627 \def\@Glsfirstplural@#1#2[#3]{%
1628   \glstrassignfieldfont{#2}%
    Ensure that \glfirstplural honours the nohyperfirst attribute.
1629   \@gls@field@link
1630   [\let\glstrifwasfirstuse\@firstoftwo
1631    \let\glstifplural\@firstoftwo
1632    \let\glscapscase\@secondofthree
1633    \glstrchecknohyperfirst{#2}%
1634   ]%
1635   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
1636 }
```

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

```
1637 \def\@GLSfirstplural@#1#2[#3]{%
1638   \glstrassignfieldfont{#2}%
    Ensure that \glfirstplural honours the nohyperfirst attribute.
1639   \@gls@field@link
1640   [\let\glstrifwasfirstuse\@firstoftwo
1641    \let\glstifplural\@firstoftwo
1642    \let\glscapscase\@thirdofthree
1643    \glstrchecknohyperfirst{#2}%
1644   ]%
1645   {#1}{#2}%
1646   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
1647 }
```

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

```
1648 \def\@glsname@#1#2[#3]{%
1649   \glstrassignfieldfont{#2}%
1650   \@gls@field@link{#1}{#2}{\@gls@field@font{\Glsaccessname{#2}#3}}%
1651 }
```

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

```
1652 \def\@Glsname@#1#2[#3]{%
1653   \glstrassignfieldfont{#2}%
1654   \@gls@field@link
1655   [\let\glscapscase\@secondoftwo]{#1}{#2}%
1656   {\@gls@field@font{\Glsaccessname{#2}#3}}%
1657 }
```

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

```
1658 \def\@GLSname@#1#2[#3]{%
1659   \glstrassignfieldfont{#2}%
1660   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1661   {#1}{#2}%
1662   {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
1663 }
```

```

\@Glsdesc@
1664 \def\@Glsdesc@#1#2[#3]{%
1665   \glstrassignfieldfont{#2}%
1666   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
1667 }

\@GLSdesc@   First letter uppercase version.
1668 \def\@GLSdesc@#1#2[#3]{%
1669   \glstrassignfieldfont{#2}%
1670   \@gls@field@link
1671   [\let\glscapscase\@secondoftwo]{#1}{#2}%
1672   {\@gls@field@font{\GLSaccessdesc{#2}#3}}%
1673 }

\@GLSdesc@   All uppercase version.
1674 \def\@GLSdesc@#1#2[#3]{%
1675   \glstrassignfieldfont{#2}%
1676   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1677   {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
1678 }

@Glsdescplural@   No case-changing version.
1679 \def\@Glsdescplural@#1#2[#3]{%
1680   \glstrassignfieldfont{#2}%
1681   \@gls@field@link
1682   [\let\glscapscase\@secondoftwo
1683    \let\glsifplural\@firstoftwo
1684   ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
1685 }

@GLSdescplural@   First letter uppercase version.
1686 \def\@GLSdescplural@#1#2[#3]{%
1687   \glstrassignfieldfont{#2}%
1688   \@gls@field@link
1689   [\let\glscapscase\@secondoftwo
1690    \let\glsifplural\@firstoftwo
1691   ]{#1}{#2}{\@gls@field@font{\GLSaccessdescplural{#2}#3}}%
1692 }

@GLSdescplural@   All uppercase version.
1693 \def\@GLSdesc@#1#2[#3]{%
1694   \glstrassignfieldfont{#2}%
1695   \@gls@field@link
1696   [\let\glscapscase\@thirdoftwo
1697   \let\glsifplural\@firstoftwo
1698   ]%
1699   {#1}{#2}%
1700   {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
1701 }

```

```

\@glssymbol@
1702 \def\@glssymbol@#1#2[#3]{%
1703   \glstrassignfieldfont{#2}%
1704   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
1705 }

\@Glsymbol@ First letter uppercase version.
1706 \def\@Glsymbol@#1#2[#3]{%
1707   \glstrassignfieldfont{#2}%
1708   \@gls@field@link
1709   [\let\glscapscase\@secondoftwo]%
1710   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
1711 }

\@GLSymbol@ All uppercase version.
1712 \def\@GLSymbol@#1#2[#3]{%
1713   \glstrassignfieldfont{#2}%
1714   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1715   {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
1716 }

lssymbolplural@ No case-changing version.
1717 \def\@lssymbolplural@#1#2[#3]{%
1718   \glstrassignfieldfont{#2}%
1719   \@gls@field@link
1720   [\let\glscapscase\@secondoftwo
1721   \let\glsifplural\@firstoftwo
1722   ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
1723 }

lssymbolplural@ First letter uppercase version.
1724 \def\@lssymbolplural@#1#2[#3]{%
1725   \glstrassignfieldfont{#2}%
1726   \@gls@field@link
1727   [\let\glscapscase\@secondoftwo
1728   \let\glsifplural\@firstoftwo
1729   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
1730 }

LSsymbolplural@ All uppercase version.
1731 \def\@LSsymbol@#1#2[#3]{%
1732   \glstrassignfieldfont{#2}%
1733   \@gls@field@link
1734   [\let\glscapscase\@thirdoftwo
1735   \let\glsifplural\@firstoftwo
1736   ]%
1737   {#1}{#2}%
1738   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
1739 }

```

\@Glsuseri@ First letter uppercase version.

```
1740 \def\@Glsuseri@#1#2[#3]{%
1741   \glstrassignfieldfont{#2}%
1742   \@gls@field@link
1743   [\let\glscapscase\@secondoftwo]{#1}{#2}%
1744   {\@gls@field@font{\Glsentryuseri{#2}#3}}%
1745 }
```

\@GLSuseri@ All uppercase version.

```
1746 \def\@GLSuseri@#1#2[#3]{%
1747   \glstrassignfieldfont{#2}%
1748   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1749   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseri{#2}#3}}}%
1750 }
```

\@Glsuserii@ First letter uppercase version.

```
1751 \def\@Glsuserii@#1#2[#3]{%
1752   \glstrassignfieldfont{#2}%
1753   \@gls@field@link
1754   [\let\glscapscase\@secondoftwo]%
1755   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
1756 }
```

\@GLSuserii@ All uppercase version.

```
1757 \def\@GLSuserii@#1#2[#3]{%
1758   \glstrassignfieldfont{#2}%
1759   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1760   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserii{#2}#3}}}%
1761 }
```

\@Glsuseriii@ First letter uppercase version.

```
1762 \def\@Glsuseriii@#1#2[#3]{%
1763   \glstrassignfieldfont{#2}%
1764   \@gls@field@link
1765   [\let\glscapscase\@secondoftwo]%
1766   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
1767 }
```

\@GLSuseriii@ All uppercase version.

```
1768 \def\@GLSuseriii@#1#2[#3]{%
1769   \glstrassignfieldfont{#2}%
1770   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1771   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
1772 }
```

\@Glsuseriv@ First letter uppercase version.

```
1773 \def\@Glsuseriv@#1#2[#3]{%
1774   \glstrassignfieldfont{#2}%

```

```

1775 \@gls@field@link
1776 [\let\glscapscase\@secondoftwo]%
1777 {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
1778 }

```

\@GLSuseriv@ All uppercase version.

```

1779 \def\@GLSuseriv@#1#2[#3]{%
1780 \glstrassignfieldfont{#2}%
1781 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1782 {#1}{#2}%
1783 {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
1784 }

```

\@Glsuserv@ First letter uppercase version.

```

1785 \def\@Glsuserv@#1#2[#3]{%
1786 \glstrassignfieldfont{#2}%
1787 \@gls@field@link
1788 [\let\glscapscase\@secondoftwo]%
1789 {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
1790 }

```

\@GLSuserv@ All uppercase version.

```

1791 \def\@GLSuserv@#1#2[#3]{%
1792 \glstrassignfieldfont{#2}%
1793 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1794 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
1795 }

```

\@Glsuservi@ First letter uppercase version.

```

1796 \def\@Glsuservi@#1#2[#3]{%
1797 \glstrassignfieldfont{#2}%
1798 \@gls@field@link
1799 [\let\glscapscase\@secondoftwo]%
1800 {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
1801 }

```

\@GLSuservi@ All uppercase version.

```

1802 \def\@GLSuservi@#1#2[#3]{%
1803 \glstrassignfieldfont{#2}%
1804 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1805 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
1806 }

```

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

\@acrshort No case change.

```

1807 \def\@acrshort#1#2[#3]{%

```



```

1808 \glsdoifexists{#2}%
1809 {%
1810   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1811   \let\glxtrifwasfirstuse\@secondoftwo
1812   \let\gl@ifplural\@secondoftwo
1813   \let\glscapscase\@firstofthree
1814   \let\glinsert\@empty
1815   \def\glscustomtext{%
1816     \acronymfont{\glaccessshort{#2}}#3%
1817   }%
1818   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1819 }%
1820 \glspostlinkhook
1821 }

```

\@Acrshort First letter uppercase.

```

1822 \def\@Acrshort#1#2[#3]{%
1823   \glsdoifexists{#2}%
1824   {%
1825     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1826     \let\glxtrifwasfirstuse\@secondoftwo
1827     \let\gl@ifplural\@secondoftwo
1828     \let\glscapscase\@secondofthree
1829     \let\glinsert\@empty
1830     \def\glscustomtext{%
1831       \acronymfont{\Glsaccessshort{#2}}#3%
1832     }%
1833     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1834   }%
1835   \glspostlinkhook
1836 }

```

\@ACRshort All uppercase.

```

1837 \def\@ACRshort#1#2[#3]{%
1838   \glsdoifexists{#2}%
1839   {%
1840     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1841     \let\glxtrifwasfirstuse\@secondoftwo
1842     \let\gl@ifplural\@secondoftwo
1843     \let\glscapscase\@thirdofthree
1844     \let\glinsert\@empty
1845     \def\glscustomtext{%
1846       \mfirstucMakeUppercase{\acronymfont{\glaccessshort{#2}}#3}%
1847     }%
1848     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1849   }%
1850   \glspostlinkhook
1851 }

```

\@acrshortpl No case change.

```
1852 \def\@acrshortpl#1#2[#3]{%
1853   \glsdoifexists{#2}%
1854   {%
1855     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1856     \let\glxtrifwasfirstuse\@secondoftwo
1857     \let\gl@sifplural\@firstoftwo
1858     \let\glscapscase\@firstofthree
1859     \let\gl$insert\@empty
1860     \def\glscustomtext{%
1861       \acronymfont{\gl@saccesssshortpl{#2}}#3%
1862     }%
1863     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1864   }%
1865   \glspostlinkhook
1866 }
```

\@Acrshortpl First letter uppercase.

```
1867 \def\@Acrshortpl#1#2[#3]{%
1868   \glsdoifexists{#2}%
1869   {%
1870     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1871     \let\glxtrifwasfirstuse\@secondoftwo
1872     \let\gl@sifplural\@firstoftwo
1873     \let\glscapscase\@secondofthree
1874     \let\gl$insert\@empty
1875     \def\glscustomtext{%
1876       \acronymfont{\Glsaccesssshortpl{#2}}#3%
1877     }%
1878     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1879   }%
1880   \glspostlinkhook
1881 }
```

\@ACRshortpl All uppercase.

```
1882 \def\@ACRshortpl#1#2[#3]{%
1883   \glsdoifexists{#2}%
1884   {%
1885     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1886     \let\glxtrifwasfirstuse\@secondoftwo
1887     \let\gl@sifplural\@firstoftwo
1888     \let\glscapscase\@thirdofthree
1889     \let\gl$insert\@empty
1890     \def\glscustomtext{%
1891       \mfirstucMakeUppercase{\acronymfont{\gl@saccesssshortpl{#2}}#3}%
1892     }%
1893     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1894   }%
1895   \glspostlinkhook
```

1896 }

\@acrlong No case change.

```
1897 \def\@acrlong#1#2[#3]{%
1898   \glsdoifexists{#2}%
1899   {%
1900     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1901     \let\glxtrifwasfirstuse\@secondoftwo
1902     \let\glsifplural\@secondoftwo
1903     \let\glscapscase\@firstofthree
1904     \let\glsinsert\@empty
1905     \def\glscustomtext{%
1906       \acronymfont{\glsaccesslong{#2}}#3%
1907     }%
1908     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1909   }%
1910   \glspostlinkhook
1911 }
```

\@Acrlong First letter uppercase.

```
1912 \def\@Acrlong#1#2[#3]{%
1913   \glsdoifexists{#2}%
1914   {%
1915     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1916     \let\glxtrifwasfirstuse\@secondoftwo
1917     \let\glsifplural\@secondoftwo
1918     \let\glscapscase\@secondofthree
1919     \let\glsinsert\@empty
1920     \def\glscustomtext{%
1921       \acronymfont{\Glsaccesslong{#2}}#3%
1922     }%
1923     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1924   }%
1925   \glspostlinkhook
1926 }
```

\@ACRlong All uppercase.

```
1927 \def\@ACRlong#1#2[#3]{%
1928   \glsdoifexists{#2}%
1929   {%
1930     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1931     \let\glxtrifwasfirstuse\@secondoftwo
1932     \let\glsifplural\@secondoftwo
1933     \let\glscapscase\@thirdofthree
1934     \let\glsinsert\@empty
1935     \def\glscustomtext{%
1936       \mfirstucMakeUppercase{\acronymfont{\glsaccesslong{#2}}#3}%
1937     }%
1938     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1939   }
```

```

1939 }%
1940 \glspostlinkhook
1941 }

```

\@acrlongpl No case change.

```

1942 \def\@acrlongpl#1#2[#3]{%
1943   \glsdoidexists{#2}%
1944   {%
1945     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1946     \let\glxtrifwasfirstuse\@secondoftwo
1947     \let\gl@sifplural\@firstoftwo
1948     \let\glscapscase\@firstofthree
1949     \let\gl$insert\@empty
1950     \def\glscustomtext{%
1951       \acronymfont{\glaccesslongpl{#2}}#3%
1952     }%
1953     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1954   }%
1955   \glspostlinkhook
1956 }

```

\@Acrlongpl First letter uppercase.

```

1957 \def\@Acrlongpl#1#2[#3]{%
1958   \glsdoidexists{#2}%
1959   {%
1960     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1961     \let\glxtrifwasfirstuse\@secondoftwo
1962     \let\gl@sifplural\@firstoftwo
1963     \let\glscapscase\@secondofthree
1964     \let\gl$insert\@empty
1965     \def\glscustomtext{%
1966       \acronymfont{\Glsaccesslongpl{#2}}#3%
1967     }%
1968     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1969   }%
1970   \glspostlinkhook
1971 }

```

\@ACRlongpl All uppercase.

```

1972 \def\@ACRlongpl#1#2[#3]{%
1973   \glsdoidexists{#2}%
1974   {%
1975     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1976     \let\glxtrifwasfirstuse\@secondoftwo
1977     \let\gl@sifplural\@firstoftwo
1978     \let\glscapscase\@thirdofthree
1979     \let\gl$insert\@empty
1980     \def\glscustomtext{%
1981       \mfirstucMakeUppercase{\acronymfont{\glaccesslongpl{#2}}#3}%

```

```

1982 }%
1983 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1984 }%
1985 \glspostlinkhook
1986 }

```

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

\@glsaddkey

```

1987 \renewcommand*{\@glsaddkey}[7]{%
1988   \key@ifundefined{glossentry}{#1}%
1989   {%
1990     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1991     \appto\@gls@keymap{,{#1}{#1}}%
1992     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
1993     \appto\@newglossaryentryposthook{%
1994       \letcs{\@glo@tmp}{@glo@#1}%
1995       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1996     }%
1997     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1998     \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

1999   \ifcsdef{@gls@user@#1@}%
2000   {%
2001     \PackageError{glossaries}%
2002     {Can't define '\string#5' as helper command
2003     '\expandafter\string\csname @gls@user@#1@\endcsname' already
2004     exists}%
2005     {}%
2006   }%
2007   {%
2008     \expandafter\newcommand\expandafter*\expandafter
2009     {\csname @gls@user@#1\endcsname}[2][]{%
2010       \new@ifnextchar[%
2011         {\csuse{@gls@user@#1@}{##1}{##2}}%
2012         {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
2013     \csdef{@gls@user@#1@}##1##2[##3]{%
2014       \@gls@field@link{##1}{##2}{#3{##2}##3}%
2015     }%
2016     \newrobustcmd*{#5}{%
2017       \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
2018   }%

```

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

```

2019   \ifcsdef{@Gls@user@#1@}%
2020   {%
2021     \PackageError{glossaries}%
2022     {Can't define '\string#6' as helper command

```

```

2023         '\expandafter\string\csname @Gls@user@#1@\endcsname' already
2024         exists}%
2025     {}%
2026 }%
2027 {%
2028     \expandafter\newcommand\expandafter*\expandafter
2029     {\csname @Gls@user@#1@\endcsname}[2][\%
2030         \new@ifnextchar[%
2031             {\csuse{@Gls@user@#1@}{##1}{##2}}}%
2032             {\csuse{@Gls@user@#1@}{##1}{##2}[]}}}%
2033     \csdef{@Gls@user@#1@}##1##2[##3]{%
2034         \@gls@field@link[\let\gls@caps@case\@secondofthree]%
2035         {##1}{##2}{#4{##2}##3}%
2036     }%
2037     \newrobustcmd*{#6}{%
2038         \expandafter\@gls@hyp@opt\csname @Gls@user@#1@\endcsname}%
2039     }%

```

Finally the all caps version (modified):

```

2040     \ifcsdef{@GLS@user@#1@}%
2041     {%
2042         \PackageError{glossaries}%
2043         {Can't define '\string#7' as helper command
2044         '\expandafter\string\csname @GLS@user@#1@\endcsname' already
2045         exists}%
2046     {}%
2047 }%
2048 {%
2049     \expandafter\newcommand\expandafter*\expandafter
2050     {\csname @GLS@user@#1@\endcsname}[2][\%
2051         \new@ifnextchar[%
2052             {\csuse{@GLS@user@#1@}{##1}{##2}}}%
2053             {\csuse{@GLS@user@#1@}{##1}{##2}[]}}}%
2054     \csdef{@GLS@user@#1@}##1##2[##3]{%
2055         \@gls@field@link[\let\gls@caps@case\@thirdofthree]%
2056         {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
2057     }%
2058     \newrobustcmd*{#7}{%
2059         \expandafter\@gls@hyp@opt\csname @GLS@user@#1@\endcsname}%
2060     }%
2061 }%
2062 {%
2063     \PackageError{glossaries-extra}{Key '#1' already exists}{}%
2064 }%
2065 }

```

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

```

2066 \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.

```
2067 \let\@glstr@org@checkfirsthyper\@gls@link@checkfirsthyper
2068 \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.

```
2069 \ifglsused{\glslabel}%
2070 {\let\glstr@ifwasfirstuse\@secondoftwo}
2071 {\let\glstr@ifwasfirstuse\@firstoftwo}%
```

Store the category label for convenience.

```
2072 \edef\glscategorylabel{\glscategory{\glslabel}}%
2073 \ifglsused{\glslabel}%
2074 {%
2075   \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
2076   {\KV@glslink@hyperfalse}{}%
2077 }%
2078 {%
2079   \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
2080   {\KV@glslink@hyperfalse}{}%
2081 }%
2082 \glslinkcheckfirsthyperhook
2083 }
```

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.

```
2084 \ifdef\do@glstdisablehyperinlist
2085 {%
2086   \let\@glstr@do@glstdisablehyperinlist\do@glstdisablehyperinlist
2087   \renewcommand*{\do@glstdisablehyperinlist}{%
2088     \@glstr@do@glstdisablehyperinlist
2089     \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}%
2090   }
2091 }
2092 }
```

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

```
2093 \define@boolkey{glslink}{noindex}[true]{}
2094 \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

```
2095 \ifdef\@gls@setdefault@glslink@opts
2096 {
2097   \renewcommand*{\@gls@setdefault@glslink@opts}{%
2098     \KV@glslink@noindexfalse
```

```

2099 \glstrsetaliasnoindex
2100 }
2101 }
2102 {
    Not defined so prepend it to \do@gl:disablehyperinlist to achieve the same effect.
2103 \newcommand*{\@gl@setdefault@gl@link@opts}{%
2104 \KV@gl@link@noindexfalse
2105 \glstrsetaliasnoindex
2106 }
2107 \pretoto\do@gl:disablehyperinlist{\@gl@setdefault@gl@link@opts}
2108 }

```

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.)

```

2109 \providecommand*\glstrsetaliasnoindex{%
2110 \KV@gl@link@noindextrue
2111 }

```

setaliasnoindex

```

2112 \newcommand*{\@glstrsetaliasnoindex}{%
2113 \ifglshasfield{alias}{\glslabel}%
2114 {%
2115 \let\glstrindexaliased\@glstrindexaliased
2116 \glstrsetaliasnoindex
2117 \let\glstrindexaliased\@no@glstrindexaliased
2118 }%
2119 }%
2120 }

```

xtrindexaliased

```

2121 \newcommand*\@glstrindexaliased{%
2122 \ifKV@gl@link@noindex
2123 \else
2124 \begin{group}
2125 \def\@gl@numberformat{gl@numberformat}%
2126 \edef\@gl@counter{\csname gl@gl@detoklabel{\glslabel}@counter\endcsname}%
2127 \glstr@saveentrycounter
2128 \do@wrglossary{\glstralias{\glslabel}}%
2129 \end{group}
2130 \fi
2131 }

```

xtrindexaliased

```

2132 \newcommand*\@no@glstrindexaliased{%
2133 \PackageError{glossaries-extra}{\string\glstrindexaliased\space
2134 not permitted outside definition of \string\glstrsetaliasnoindex}%
2135 {}%
2136 }

```



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

```
2137 \let\glxtrindexaliased\@no\glxtrindexaliased
```

tDefaultGlsOpts Set the default options for \glslink etc.

```
2138 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
2139   \renewcommand*{\@glslsetdefault@glslink@opts}{%
2140     \setkeys{glslink}{#1}%
2141     \@glxtrsetaliasnoindex
2142   }%
2143 }
```

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

```
2144 \newcommand*{\glxtrifindexing}[2]{%
2145   \ifKV@glslink@noindex #2\else #1\fi
2146 }
```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```
2147 \renewcommand*{\glswriteentry}[2]{%
2148   \glxtrifindexing
2149   {%
2150     \ifglindexonlyfirst
2151       \ifglused{#1}
2152       {\glxtrdoautoindexname{#1}{dualindex}}%
2153       {#2}%
2154     \else
2155       \gl@ifattribute{#1}{indexonlyfirst}{true}%
2156       {\ifglused{#1}
2157        {\glxtrdoautoindexname{#1}{dualindex}}%
2158        {#2}}%
2159       {#2}%
2160     \fi
2161   }%
2162   {}%
2163 }
```

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

```
2164 \appto\@do@wrglossary{\@glxtr@do@wrindex
2165   \glxtrdowrglossaryhook{\@glsl@label}%
2166 }
```

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

Similarly for the “noidx” version:

s@noidxglossary

```
2167 \appto\glsl@noidxglossary{\@glxtr@do@wrindex
2168   \glxtrdowrglossaryhook{\@glsl@label}%
2169 }
```

xtr@do@@wrindex

```
2170 \newcommand*{\@glsxtr@do@@wrindex}{%
2171   \glsxtrdoautoindexname{\@gls@label}{dualindex}%
2172 }
```

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.)

```
2173 \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.

```
2174 \newcommand*{\@gls@alt@hyp@opt}[1]{%
2175   \let\glslinkvar\@firstofthree
2176   \let\@gls@hyp@opt@cs#1\relax
2177   \@ifstar{\s@gls@hyp@opt}%
2178   {\@ifnextchar+%
2179     {\@firstoftwo{\p@gls@hyp@opt}}%
2180     {%
2181       \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
2182       {\@firstoftwo{\@alt@gls@hyp@opt}}%
2183       {#1}%
2184     }%
2185   }%
2186 }
```

alt@gls@hyp@opt User version

```
2187 \newcommand*{\@alt@gls@hyp@opt}[1] [] {%
2188   \let\glslinkvar\@firstofthree
2189   \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

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

```
2190 \newcommand*{\@gls@alt@hyp@opt@char}{{}
```

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

```
2191 \newcommand*{\@gls@alt@hyp@opt@keys}{{}
```

rSetAltModifier

```
2192 \newcommand*{\GlsXtrSetAltModifier}[2]{%
2193   \let\@gls@hyp@opt\@gls@alt@hyp@opt
2194   \def\@gls@alt@hyp@opt@char{#1}%
2195   \def\@gls@alt@hyp@opt@keys{#2}%
2196 }
```

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

or using the plus version.) This also patches the short form commands like `\acrshort` and `\glxtrshort` to use `\glsentryshort` and, similarly, the long form commands like `\acrlong` and `\glxtrlong` to use `\glsentrylong`. Added attribute check.

```

2197 \renewcommand*{\glsdohyperlink}[2]{%
2198   \glsbasattribute{\glslabel}{targeturl}%
2199   {%
2200     \glsbasattribute{\glslabel}{targetname}%
2201     {%
2202       \glsbasattribute{\glslabel}{targetcategory}%
2203       {%
2204         \hyperref{\glsgetattribute{\glslabel}{targeturl}}{%
2205           {\glsgetattribute{\glslabel}{targetcategory}}}%
2206           {\glsgetattribute{\glslabel}{targetname}}}%
2207           {\{\glsxtrprotectlinks#2}\}}%
2208         }%
2209       }%
2210       \hyperref{\glsgetattribute{\glslabel}{targeturl}}{%
2211         }%
2212         {\glsgetattribute{\glslabel}{targetname}}}%
2213         {\{\glsxtrprotectlinks#2}\}}%
2214       }%
2215     }%
2216   }%
2217   \href{\glsgetattribute{\glslabel}{targeturl}}{%
2218     {\{\glsxtrprotectlinks#2}\}}%
2219   }%
2220 }%
2221 {%

```

Check for alias.

```

2222   \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
2223   \ifvoid\gloaliaslabel
2224   {%
2225     \hyperlink{#1}{\{\glsxtrprotectlinks#2}\}}%
2226   }%
2227   {%

```

Redirect link to the alias target.

```

2228     \hyperlink
2229     {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
2230     {\{\glsxtrprotectlinks#2}\}}%
2231   }%
2232 }%
2233 }

```

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

```

2234 \renewrobustcmd*{\gls hyperlink}[2][\glsentrytext{\@glo@label}]{%
2235   \def\@glo@label{#2}%
2236   {\edef\glslabel{#2}%
2237    \@glslink{\glo@linkprefix\glslabel}{#1}}%
2238 }

```

`glsdisablehyper`   Redefine in case we have an old version of glossaries.

```

2239 \ifundef\glsdonohyperlink
2240 {%
2241   \renewcommand{\glsdisablehyper}{%
2242     \KV@glslink@hyperfalse
2243     \let\@glslink\glsdonohyperlink
2244     \let\@glstarget\@secondoftwo
2245   }
2246 }
2247 {}

```

`glsdonohyperlink`   This command was only introduced in glossaries v4.20, so it may not be defined. 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.

```

2248 \def\glsdonohyperlink#1#2{{\glsxtrprotectlinks #2}}

```

Reset `\@glslink` with patched versions:

```

2249 \ifcsundef{hyperlink}%
2250 {%
2251   \let\@glslink\glsdonohyperlink
2252 }%
2253 {%
2254   \let\@glslink\glsdohyperlink
2255 }

```

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

```

2256 \newcommand*{\glsxtrprotectlinks}{%
2257   \KV@glslink@hyperfalse
2258   \KV@glslink@noindextrue
2259   \let\@gls@\@glsxtr@p@text@
2260   \let\@Gls@\@Glsxtr@p@text@
2261   \let\@GLS@\@GLSxtr@p@text@
2262   \let\@glspl@\@glsxtr@p@plural@
2263   \let\@Glspl@\@Glsxtr@p@plural@
2264   \let\@GLSpl@\@GLSxtr@p@plural@
2265   \let\@glsxtrshort@\@glsxtr@p@short@
2266   \let\@Glsxtrshort@\@Glsxtr@p@short@
2267   \let\@GLSxtrshort@\@GLSxtr@p@short@
2268   \let\@glsxtrlong@\@glsxtr@p@long@
2269   \let\@Glsxtrlong@\@Glsxtr@p@long@
2270   \let\@GLSxtrlong@\@GLSxtr@p@long@

```

```

2271 \let\@glsxtrshortpl\@glsxtr@p@shortpl@
2272 \let\@Glsxtrshortpl\@Glsxtr@p@shortpl@
2273 \let\@GLSxtrshortpl\@GLSxtr@p@shortpl@
2274 \let\@glsxtrlongpl\@glsxtr@p@longpl@
2275 \let\@Glsxtrlongpl\@Glsxtr@p@longpl@
2276 \let\@GLSxtrlongpl\@GLSxtr@p@longpl@
2277 \let\@acrshort\@glsxtr@p@acrshort@
2278 \let\@Acrshort\@Glsxtr@p@acrshort@
2279 \let\@ACRshort\@GLSxtr@p@acrshort@
2280 \let\@acrshortpl\@glsxtr@p@acrshortpl@
2281 \let\@Acrshortpl\@Glsxtr@p@acrshortpl@
2282 \let\@ACRshortpl\@GLSxtr@p@acrshortpl@
2283 \let\@acrlong\@glsxtr@p@acrlong@
2284 \let\@Acrlong\@Glsxtr@p@acrlong@
2285 \let\@ACRlong\@GLSxtr@p@acrlong@
2286 \let\@acrlongpl\@glsxtr@p@acrlongpl@
2287 \let\@Acrlongpl\@Glsxtr@p@acrlongpl@
2288 \let\@ACRlongpl\@GLSxtr@p@acrlongpl@
2289 }

```

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

@glsxtr@p@text@

```
2290 \def\@glsxtr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}
```

@Glsxtr@p@text@

```
2291 \def\@Glsxtr@p@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}
```

@GLSxtr@p@text@

```
2292 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}
```

lsxtr@p@plural@

```
2293 \def\@glsxtr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}
```

Glsxtr@p@plural@

```
2294 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}
```

LSxtr@p@plural@

```
2295 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}
```

glsxtr@p@short@

```

2296 \def\@glsxtr@p@short@#1#2[#3]{%
2297   {%
2298     \glssetabbrvfmt{\glscategory{#2}}%
2299     \glsabbrvfont{\glsentryshort{#2}}#3%
2300   }%
2301 }

```

Glsxtr@p@short@

```
2302 \def\@Glsxtr@p@short@#1#2[#3]{%
2303   {%
2304     \glsetabbrvfmt{\glscategory{#2}}%
2305     \glabbrvfont{\Glsentryshort{#2}}#3%
2306   }%
2307 }
```

GLSxtr@p@short@

```
2308 \def\@GLSxtr@p@short@#1#2[#3]{%
2309   {%
2310     \glsetabbrvfmt{\glscategory{#2}}%
2311     \mfirstucMakeUppercase{\glabbrvfont{\Glsentryshort{#2}}#3}%
2312   }%
2313 }
```

sxtr@p@shortpl@

```
2314 \def\@glxtr@p@shortpl@#1#2[#3]{%
2315   {%
2316     \glsetabbrvfmt{\glscategory{#2}}%
2317     \glabbrvfont{\Glsentryshortpl{#2}}#3%
2318   }%
2319 }
```

sxtr@p@shortpl@

```
2320 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
2321   {%
2322     \glsetabbrvfmt{\glscategory{#2}}%
2323     \glabbrvfont{\Glsentryshortpl{#2}}#3%
2324   }%
2325 }
```

Sxtr@p@shortpl@

```
2326 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
2327   {%
2328     \glsetabbrvfmt{\glscategory{#2}}%
2329     \mfirstucMakeUppercase{\glabbrvfont{\Glsentryshortpl{#2}}#3}%
2330   }%
2331 }
```

@glxtr@p@long@

```
2332 \def\@glxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}#3}}
```

@Glsxtr@p@long@

```
2333 \def\@Glsxtr@p@long@#1#2[#3]{\{\Glsentrylong{#2}#3}}
```

@GLSxtr@p@long@

```
2334 \def\@GLSxtr@p@long@#1#2[#3]{%
2335   {\mfirstucMakeUppercase{\glslongfont{\Glsentrylong{#2}}#3}}}
```

lsxtr@p@longpl@  
2336 \def\@glxtr@p@longpl@#1#2[#3]{\glentrylongpl{#2}#3}}

lsxtr@p@longpl@  
2337 \def\@Glsxtr@p@longpl@#1#2[#3]{\glslongfont{\Glentrylongpl{#2}#3}}

LSxtr@p@longpl@  
2338 \def\@GLSxtr@p@longpl@#1#2[#3]{%  
2339 {\mfirstucMakeUppercase{\glslongfont{\glentrylongpl{#2}#3}}}

xtr@p@acrshort@  
2340 \def\@glxtr@p@acrshort@#1#2[#3]{\acronymfont{\glentryshort{#2}#3}}

xtr@p@acrshort@  
2341 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glentryshort{#2}#3}}

xtr@p@acrshort@  
2342 \def\@GLSxtr@p@acrshort@#1#2[#3]{%  
2343 {\mfirstucMakeUppercase{\acronymfont{\glentryshort{#2}#3}}}

r@p@acrshortpl@  
2344 \def\@glxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glentryshortpl{#2}#3}}

r@p@acrshortpl@  
2345 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glentryshortpl{#2}#3}}

r@p@acrshortpl@  
2346 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%  
2347 {\mfirstucMakeUppercase{\acronymfont{\glentryshortpl{#2}#3}}}

sxtr@p@acrlong@  
2348 \def\@glxtr@p@acrlong@#1#2[#3]{\glentrylong{#2}#3}}

sxtr@p@acrlong@  
2349 \def\@Glsxtr@p@acrlong@#1#2[#3]{\Glentrylong{#2}#3}}

Sxtr@p@acrlong@  
2350 \def\@GLSxtr@p@acrlong@#1#2[#3]{%  
2351 {\mfirstucMakeUppercase{\glentrylong{#2}#3}}}

tr@p@acrlongpl@  
2352 \def\@glxtr@p@acrlongpl@#1#2[#3]{\glentrylongpl{#2}#3}}

tr@p@acrlongpl@  
2353 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\Glentrylongpl{#2}#3}}

tr@p@acrlongpl@

```
2354 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
2355 {\mfirstucMakeUppercase{\glstrylongpl{#2}#3}}}
```

Commands to minimise conflict.

\@glxtrp@opt

```
2356 \newcommand*{\@glxtrp@opt}{hyper=false,noindex}
```

\glxtrsetopts Used in glossary to switch hyperlinks on for the \@glxtrp type of commands.

```
2357 \newcommand*{\glxtrsetopts}[1]{%
2358 \renewcommand*{\@glxtrp@opt}{#1}%
2359 }
```

lossxtrsetopts Used in glossary to switch hyperlinks on for the \@glxtrp type of commands.

```
2360 \newcommand*{\glossxtrsetopts}{%
2361 \glxtrsetopts{noindex}%
2362 }
```

\@@glxtrp

```
2363 \newrobustcmd*{\@@glxtrp}[2]{%
```

Add scope.

```
2364 {%
2365 \let\glspostlinkhook\relax
2366 \csname#1\expandafter\endcsname\expandafter[\@glxtrp@opt]{#2}[]%
2367 }%
2368 }
```

\@glxtrp

```
2369 \newrobustcmd*{\@glxtrp}[2]{%
2370 \ifcsdef{gls#1}%
2371 {%
2372 \@glxtrp{gls#1}{#2}%
2373 }%
2374 {%
2375 \ifcsdef{glsxtr#1}%
2376 {%
2377 \@glxtrp{glsxtr#1}{#2}%
2378 }%
2379 {%
2380 \PackageError{glossaries-extra}{‘#1’ not recognised by
2381 \string\glxtrp}{}%
2382 }%
2383 }%
2384 }
```

\@Glsxtrp

```
2385 \newrobustcmd*{\@Glsxtrp}[2]{%
```



```

2386 \ifcsdef{Gls#1}%
2387 {%
2388   \@@glxstrp{Gls#1}{#2}%
2389 }%
2390 {%
2391   \ifcsdef{Glsxtr#1}%
2392   {%
2393     \@@glxstrp{Glsxtr#1}{#2}%
2394   }%
2395   {%
2396     \PackageError{glossaries-extra}{‘#1’ not recognised by
2397       \string\Glsxtrp}{}%
2398   }%
2399 }%
2400 }

```

\@GLSxtrp

```

2401 \newrobustcmd*{\@GLSxtrp}[2]{%
2402   \ifcsdef{GLS#1}%
2403   {%
2404     \@@glxstrp{GLS#1}{#2}%
2405   }%
2406   {%
2407     \ifcsdef{GLSxtr#1}%
2408     {%
2409       \@@glxstrp{GLSxtr#1}{#2}%
2410     }%
2411     {%
2412       \PackageError{glossaries-extra}{‘#1’ not recognised by
2413         \string\GLSxtrp}{}%
2414     }%
2415   }%
2416 }

```

\glxstr@entry@p

```

2417 \newrobustcmd*{\glxstr@headentry@p}[2]{%
2418   \gl@ifattribute{#1}{headuc}{true}%
2419   {%
2420     \mfirstucMakeUppercase{\@gl@entry@field{#1}{#2}}%
2421   }%
2422   {%
2423     \@gl@entry@field{#1}{#2}%
2424   }%
2425 }

```

\glxstrp Not robust as it needs to expand somewhat.

```

2426 \ifdef\texorpdfstring
2427 {
2428   \newcommand{\glxstrp}[2]{%

```

```

2429 \protect\NoCaseChange
2430 {%
2431 \protect\texorpdfstring
2432 {%
2433 \protect\glstrifinmark
2434 {%
2435 \ifcsdef{glstrhead#1}%
2436 {%
2437 {\protect\csuse{glstrhead#1}{#2}}%
2438 }%
2439 {%
2440 \glstr@headentry@p{#2}{#1}%
2441 }%
2442 }%
2443 {%
2444 \@glstrp{#1}{#2}%
2445 }%
2446 }%
2447 {%
2448 \protect\@glstr@entry@field{#2}{#1}%
2449 }%
2450 }%
2451 }
2452 }
2453 {
2454 \newcommand{\glstrp}[2]{%
2455 \protect\NoCaseChange
2456 {%
2457 \protect\glstrifinmark
2458 {%
2459 \ifcsdef{glstrhead#1}%
2460 {%
2461 {\protect\csuse{glstrhead#1}}%
2462 }%
2463 {%
2464 \glstr@headentry@p{#2}{#1}%
2465 }%
2466 }%
2467 {%
2468 \@glstrp{#1}{#2}%
2469 }%
2470 }%
2471 }
2472 }

```

Provide short synonyms for the most common option.

`\glsp`

```

2473 \newcommand*{\glsp}{\glstrp{short}}

```

`\glspt`

```
2474 \newcommand*{\glspt}{\glsxtrp{text}}
```

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

```
2475 \ifdef\texorpdfstring
2476 {
2477   \newcommand{\Glsxtrp}[2]{%
2478     \protect\NoCaseChange
2479     {%
2480       \protect\texorpdfstring
2481       {%
2482         \protect\glsxtrifinmark
2483         {%
2484           \ifcsdef{Glsxtrhead#1}%
2485           {%
2486             {\protect\csuse{Glsxtrhead#1}{#2}}%
2487           }%
2488           {%
2489             \protect\@Gls@entry@field{#2}{#1}%
2490           }%
2491         }%
2492         {%
2493           \@Glsxtrp{#1}{#2}%
2494         }%
2495       }%
2496       {%
2497         \protect\@gls@entry@field{#2}{#1}%
2498       }%
2499     }%
2500   }
2501 }
2502 {
2503   \newcommand{\Glsxtrp}[2]{%
2504     \protect\NoCaseChange
2505     {%
2506       \protect\glsxtrifinmark
2507       {%
2508         \ifcsdef{Glsxtrhead#1}%
2509         {%
2510           {\protect\csuse{Glsxtrhead#1}}%
2511         }%
2512         {%
2513           \protect\@Gls@entry@field{#2}{#1}%
2514         }%
2515       }%
2516       {%
2517         \@Glsxtrp{#1}{#2}%
2518       }%
2519     }
2520 }
```

```

2519     }%
2520   }
2521 }

```

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

```

2522 \ifdef\teorpdfstring
2523 {
2524   \newcommand{\GLSxtrp}[2]{%
2525     \protect\NoCaseChange
2526     {%
2527       \protect\teorpdfstring
2528       {%
2529         \protect\glxtrifinmark
2530         {%
2531           \ifcsdef{GLSxtr#1}%
2532           {%
2533             {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
2534           }%
2535           {%
2536             \protect\mfirstucMakeUppercase
2537             {%
2538               \protect\@gls@entry@field{#2}{#1}%
2539             }%
2540           }%
2541         }%
2542       }%
2543       \@GLSxtrp{#1}{#2}%
2544     }%
2545   }%
2546   {%
2547     \protect\@gls@entry@field{#2}{#1}%
2548   }%
2549 }%
2550 }
2551 }
2552 {
2553   \newcommand{\GLSxtrp}[2]{%
2554     \protect\NoCaseChange
2555     {%
2556       \protect\glxtrifinmark
2557       {%
2558         \ifcsdef{GLSxtr#1}%
2559         {%
2560           {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
2561         }%
2562         {%
2563           \protect\mfirstucMakeUppercase
2564           {%
2565             \protect\@gls@entry@field{#2}{#1}%

```

```

2566         }%
2567     }%
2568 }%
2569 {%
2570     \@GLSxtrp{#1}{#2}%
2571 }%
2572 }%
2573 }
2574 }

```

### 1.3.5 Entry Counting

The entry counting mechanism from glossaries is adjusted here to work with category attributes. Provide a convenient command to enable entry counting, set the entrycount attribute for given categories and redefine \gls etc to use \cgl s instead.

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

\@glsunset Global unset.

```

2575 \renewcommand*{\@glsunset}[1]{%
2576     \@glsunset{#1}%
2577     \glsxtrpostunset{#1}%
2578 }%

```

glsxtrpostunset

```

2579 \newcommand*{\glsxtrpostunset}[1]{%

```

\@glslocalunset Local unset.

```

2580 \renewcommand*{\@glslocalunset}[1]{%
2581     \@glslocalunset{#1}%
2582     \glsxtrpostlocalunset{#1}%
2583 }%

```

rpostlocalunset

```

2584 \newcommand*{\glsxtrpostlocalunset}[1]{%

```

\@glsreset Global reset.

```

2585 \renewcommand*{\@glsreset}[1]{%
2586     \@glsreset{#1}%
2587     \glsxtrpostreset{#1}%
2588 }%

```

glsxtrpostreset

```

2589 \newcommand*{\glsxtrpostreset}[1]{%

```

\@glslocalreset Local reset.

```

2590 \renewcommand*{\@glslocalreset}[1]{%
2591     \@glslocalreset{#1}%
2592     \glsxtrpostlocalreset{#1}%
2593 }%

```

rpostlocalreset

```
2594 \newcommand*{\glxtrpostlocalreset}[1]{}
```

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

```
2595 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

Enable entry counting:

```
2596 \glsenableentrycount
```

Redefine \gls etc:

```
2597 \renewcommand*{\gls}{\cglss}%
2598 \renewcommand*{\Gls}{\cGls}%
2599 \renewcommand*{\glspl}{\cglsspl}%
2600 \renewcommand*{\Glspl}{\cGlspl}%
2601 \renewcommand*{\GLS}{\cGLS}%
2602 \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
2603 \@glxtr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
2604 \let\GlsXtrEnableEntryCounting\@glxtr@setentrycountunsetattr
2605 \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
2606 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
2607 can't be used with \string\GlsXtrEnableEntryCounting}%
2608 {Use one or other but not both commands}}%
2609 }
```

ycountunsetattr

```
2610 \newcommand*{\@glxtr@setentrycountunsetattr}[2]{%
2611 \@for\@glxtr@cat:=#1\do
2612 {%
2613 \ifdefempty{\@glxtr@cat}{}%
2614 {%
2615 \glsssetcategoryattribute{\@glxtr@cat}{entrycount}{#2}%
2616 }%
2617 }%
2618 }
```

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

nableentrycount

```
2619 \renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
2620 \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defcounters}%
```

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

```
2621 \renewcommand*{\gls@defdocnewglossaryentry}{%
2622 \renewcommand*\newglossaryentry[2]{%
```

```

2623 \PackageError{glossaries}{\string\newglossaryentry\space
2624 may only be used in the preamble when entry counting has
2625 been activated}{If you use \string\glsenableentrycount\space
2626 you must place all entry definitions in the preamble not in
2627 the document environment}%
2628 }%
2629 }%

```

New commands to access new fields:

```

2630 \newcommand*{\glsentrycurrcount}[1]{%
2631 \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2632 {0}{\@gls@entry@field{##1}{currcount}}}%
2633 }%
2634 \newcommand*{\glsentryprevcount}[1]{%
2635 \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2636 {0}{\@gls@entry@field{##1}{prevcount}}}%
2637 }%

```

Adjust post unset and reset:

```

2638 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
2639 \renewcommand*{\glsxtrpostunset}[1]{%
2640 \@glsxtr@entrycount@org@unset{##1}%
2641 \@gls@increment@currcount{##1}%
2642 }%
2643 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
2644 \renewcommand*{\glsxtrpostlocalunset}[1]{%
2645 \@glsxtr@entrycount@org@localunset{##1}%
2646 \@gls@local@increment@currcount{##1}%
2647 }%
2648 \let\@glsxtr@entrycount@org@reset\glsxtrpostreset
2649 \renewcommand*{\glsxtrpostreset}[1]{%
2650 \@glsxtr@entrycount@org@reset{##1}%
2651 \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2652 }%
2653 \let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
2654 \renewcommand*{\glsxtrpostlocalreset}[1]{%
2655 \@glsxtr@entrycount@org@localreset{##1}%
2656 \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2657 }%

```

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

```

2658 \let\@cgl@s@\@cgl@s@
2659 \let\@cgl spl@\@cgl spl@

2660 \let\@cGls@\@cGls@
2661 \let\@cGlspl@\@cGlspl@
2662 \let\@cGLS@\@cGLS@
2663 \let\@cGLSpl@\@cGLSpl@

```

The rest is as the original definition.

```

2664 \AtEndDocument{\@gls@write@entrycounts}%
2665 \renewcommand*{\@gls@entry@count}[2]{%
2666   \csgdef{glo@glsetoklabel{##1}@prevcount}{##2}%
2667 }%
2668 \let\glsenableentrycount\relax
2669 \renewcommand*{\glsenableentryunitcount}{%
2670   \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
2671     can't be used with \string\glsenableentrycount}%
2672   {Use one or other but not both commands}%
2673 }%
2674 }

```

`ite@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.

```

2675 \renewcommand*{\@gls@write@entrycounts}{%
2676   \immediate\write\@auxout
2677   {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2678   \count@=0\relax
2679   \forallglsentries{\@glsentry}{%
2680     \glshasattribute{\@glsentry}{entrycount}%
2681     {%
2682       \ifglused{\@glsentry}%
2683       {%
2684         \immediate\write\@auxout
2685         {\string\@gls@entry@count{\@glsentry}{\glentrycurrcount{\@glsentry}}}%
2686       }%
2687     }%
2688     \advance\count@ by \@ne
2689   }%
2690 }%
2691 }%
2692 \ifnum\count@=0
2693   \GlossariesExtraWarningNoLine{Entry counting has been enabled
2694     \MessageBreak with \string\glsenableentrycount\space but the
2695     \MessageBreak attribute 'entrycount' hasn't
2696     \MessageBreak been assigned to any of the defined
2697     \MessageBreak entries}%
2698 \fi
2699 }

```

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

```

2700 \newcommand*{\glxtrifcounttrigger}[3]{%
2701   \glshasattribute{#1}{entrycount}%
2702   {%
2703     \ifnum\glentryprevcount{#1}>\glsetattribute{#1}{entrycount}\relax

```



```

2704     #3%
2705   \else
2706     #2%
2707   \fi
2708 }%
2709 {#3}%
2710 }

```

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

\@@cgl@

```

2711 \def\@@cgl@#1#2[#3]{%
2712   \glxtrifcounttrigger{#2}%
2713   {%
2714     \cglformat{#2}{#3}%
2715     \glunset{#2}%
2716   }%
2717   {%
2718     \@gls@{#1}{#2}[#3]%
2719   }%
2720 }%

```

\@@cglsp@

```

2721 \def\@@cglsp@#1#2[#3]{%
2722   \glxtrifcounttrigger{#2}%
2723   {%
2724     \cglspformat{#2}{#3}%
2725     \glunset{#2}%
2726   }%
2727   {%
2728     \@glsp@{#1}{#2}[#3]%
2729   }%
2730 }%

```

\@@cGls@

```

2731 \def\@@cGls@#1#2[#3]{%
2732   \glxtrifcounttrigger{#2}%
2733   {%
2734     \cGlsformat{#2}{#3}%
2735     \glunset{#2}%
2736   }%
2737   {%
2738     \@Gls@{#1}{#2}[#3]%
2739   }%
2740 }%

```

\@@cGlspl@

```

2741 \def\@@cGlspl@#1#2[#3]{%
2742   \glxtrifcounttrigger{#2}%

```

```

2743 {%
2744   \cGlsplformat{#2}{#3}%
2745   \glsunset{#2}%
2746 }%
2747 {%
2748   \@Glspl@{#1}{#2}[#3]%
2749 }%
2750 }%

```

\@@cGLS@

```

2751 \def\@@cGLS@#1#2[#3]{%
2752   \glsxtrifcounttrigger{#2}%
2753   {%
2754     \cGLSformat{#2}{#3}%
2755     \glsunset{#2}%
2756   }%
2757   {%
2758     \@GLS@{#1}{#2}[#3]%
2759   }%
2760 }%

```

\@@cGLSpl@

```

2761 \def\@@cGLSpl@#1#2[#3]{%
2762   \glsxtrifcounttrigger{#2}%
2763   {%
2764     \cGLSplformat{#2}{#3}%
2765     \glsunset{#2}%
2766   }%
2767   {%
2768     \@GLSpl@{#1}{#2}[#3]%
2769   }%
2770 }%

```

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

\@cgl's@

```

2771 \def\@cgl's@#1#2[#3]{\@gls@{#1}{#2}[#3]}

```

\@cGls@

```

2772 \def\@cGls@#1#2[#3]{\@Gls@{#1}{#2}[#3]}

```

\@cgl'spl@

```

2773 \def\@cgl'spl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}

```

\@cGlspl@

```

2774 \def\@cGlspl@#1#2[#3]{\@Glspl@{#1}{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

\cGLS

```
2775 \newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}
```

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

```
2776 \newcommand*{\@cGLS}[2] [] {%
2777   \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2} []}%
2778 }
```

\@cGLS@

```
2779 \def\@cGLS@#1#2[#3]{\@cGLS@{#1}{#2}[#3]}
```

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

```
2780 \newcommand*{\cGLSformat}[2] {%
2781   \expandafter\mfirstuc\MakeUppercase\expandafter{\cglformat{#1}{#2}}%
2782 }
```

\cGLSpl

```
2783 \newrobustcmd*{\cGLSpl}{\@gls@hyp@opt\@cGLSpl}
```

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

```
2784 \newcommand*{\@cGLSpl}[2] [] {%
2785   \new@ifnextchar[{\@cGLSpl@{#1}{#2}}{\@cGLSpl@{#1}{#2} []}%
2786 }
```

\@cGLSpl@

```
2787 \def\@cGLSpl@#1#2[#3]{\@cGLSpl@{#1}{#2}[#3]}
```

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

```
2788 \newcommand*{\cGLSplformat}[2] {%
2789   \expandafter\mfirstuc\MakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
2790 }
```

Modify the trigger formats to check for the regular attribute.

\cglformat

```
2791 \renewcommand*{\cglformat}[2] {%
2792   \glsifregular{#1}
2793   {\glsentryfirst{#1}}%
2794   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}%#2%
2795 }
```

\cGlsformat

```
2796 \renewcommand*{\cGlsformat}[2] {%
2797   \glsifregular{#1}
2798   {\Glsentryfirst{#1}}%
2799   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}%#2%
2800 }
```

\cglsplformat

```
2801 \renewcommand*{\cglsplformat}[2]{%
2802   \glsifregular{#1}
2803   {\glsentryfirstplural{#1}}%
2804   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}\#2%
2805 }
```

\cGlsplformat

```
2806 \renewcommand*{\cGlsplformat}[2]{%
2807   \glsifregular{#1}
2808   {\Glsentryfirstplural{#1}}%
2809   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}\#2%
2810 }
```

New code similar to above for unit counting.

defunitcounters

```
2811 \newcommand*{\@@newglossaryentry@defunitcounters}{%
2812   \edef\@glo@countunit{\csuse{@glxtr@categoryattr@@\@glo@category @unitcount}}%
2813   \ifdefvoid\@glo@countunit
2814   {}%
2815   {%
2816     \@glxtr@ifunitcounter{\@glo@countunit}%
2817     {}%
2818     {\expandafter\@glxtr@addunitcounter\expandafter{\@glo@countunit}}%
2819   }%
2820 }
```

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

```
2821 \newcommand*{\@glxtr@unitcountlist}{}%
```

@addunitcounter

```
2822 \newcommand*{\@glxtr@addunitcounter}[1]{%
2823   \listadd{\@glxtr@unitcountlist}{#1}%
2824   \ifcsundef{glxtr@theunit@#1}
2825   {%
2826     \ifcsdef{theH#1}%
2827     {\csdef{glxtr@theunit@#1}{\csuse{theH#1}}}%
2828     {\csdef{glxtr@theunit@#1}{\csuse{the#1}}}%
2829   }%
2830   {}%
2831 }
```

r@ifunitcounter

```
2832 \newcommand*{\@glxtr@ifunitcounter}[3]{%
2833   \xifinlist{#1}{\@glxtr@unitcountlist}{#2}{#3}%
2834 }
```

urrentunitcount

```
2835 \newcommand*\@glxtr@currentunitcount[1]{%
2836   glo@\glstoklabel{#1}@currunit@\glstgetattribute{#1}{unitcount}.%
2837   \csuse{glxtr@theunit@\glstgetattribute{#1}{unitcount}}%
2838 }
```

previousunitcount

```
2839 \newcommand*\@glxtr@previousunitcount[1]{%
2840   glo@\glstoklabel{#1}@prevunit@\glstgetattribute{#1}{unitcount}.%
2841   \csuse{glxtr@theunit@\glstgetattribute{#1}{unitcount}}%
2842 }
```

t@currunitcount

```
2843 \newcommand*\@glst@increment@currunitcount[1]{%
2844   \glshasattribute{#1}{unitcount}%
2845   {%
2846     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
2847     \ifcsundef{\@glxtr@csname}%
2848     {%
2849       \csgdef{\@glxtr@csname}{1}%
2850       \listcsxadd
2851         {glo@\glstoklabel{#1}@unitlist}%
2852         {\glstgetattribute{#1}{unitcount}.%
2853         \csuse{glxtr@theunit@\glstgetattribute{#1}{unitcount}}%
2854         }%
2855     }%
2856     {%
2857       \csxdef{\@glxtr@csname}%
2858       {\number\numexpr\csname\@glxtr@csname\endcsname+1}%
2859     }%
2860   }%
2861   {}%
2862 }
```

t@currunitcount

```
2863 \newcommand*\@glst@local@increment@currunitcount[1]{%
2864   \glshasattribute{#1}{unitcount}%
2865   {%
2866     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
2867     \ifcsundef{\@glxtr@csname}%
2868     {%
2869       \csdef{\@glxtr@csname}{1}%
2870       \listcseadd
2871         {glo@\glstoklabel{#1}@unitlist}%
2872         {\glstgetattribute{#1}{unitcount}.%
2873         \csuse{glxtr@theunit@\glstgetattribute{#1}{unitcount}}%
2874         }%
2875     }%
2876     {%
```

```

2877 \csedef{\@glsxtr@csname}%
2878 {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
2879 }%
2880 }%
2881 {}%
2882 }

```

r@currunitcount

```

2883 \newcommand*{\@glsxtr@currunitcount}[2]{%
2884 \ifcsundef
2885 {glo@\glsdetoklabel{#1}@currunit@#2}%
2886 {0}%
2887 {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
2888 }%

```

r@prevunitcount

```

2889 \newcommand*{\@glsxtr@prevunitcount}[2]{%
2890 \ifcsundef
2891 {glo@\glsdetoklabel{#1}@prevunit@#2}%
2892 {0}%
2893 {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}%
2894 }%

```

eentryunitcount

```

2895 \newcommand*{\glsenableentryunitcount}{%
  Enable new fields:
2896 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
  Just in case the user has switched on the docdef option.
2897 \renewcommand*{\gls@defdocnewglossaryentry}{%
2898 \renewcommand*\newglossaryentry[2]{%
2899 \PackageError{glossaries}{\string\newglossaryentry\space
2900 may only be used in the preamble when entry counting has
2901 been activated}{If you use \string\glsenableentryunitcount\space
2902 you must place all entry definitions in the preamble not in
2903 the document environment}%
2904 }%
2905 }%

```

New commands to access new fields:

```

2906 \newcommand*{\glsentrycurrcount}[1]{%
2907 \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
2908 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
2909 }%
2910 \newcommand*{\glsentryprevcount}[1]{%
2911 \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
2912 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
2913 }%

```

Access total count:

```

2914 \newcommand*{\glstentryprevtotalcount}[1]{%
2915   \ifcsundef{glo@\glstetoklabel{##1}@prevunittotal}%
2916     {0}%
2917     {%
2918       \number\csuse{glo@\glstetoklabel{##1}@prevunittotal}
2919     }%
2920 }%

```

Access max value:

```

2921 \newcommand*{\glstentryprevmaxcount}[1]{%
2922   \ifcsundef{glo@\glstetoklabel{##1}@prevunitmax}%
2923     {0}%
2924     {%
2925       \number\csuse{glo@\glstetoklabel{##1}@prevunitmax}
2926     }%
2927 }%

```

Adjust post unset and reset:

```

2928 \let\@glstxtr@entryunitcount@org@unset\glstxtrpostunset
2929 \renewcommand*{\glstxtrpostunset}[1]{%
2930   \@glstxtr@entryunitcount@org@unset{##1}%
2931   \@glst@increment@currunitcount{##1}%
2932 }%
2933 \let\@glstxtr@entryunitcount@org@localunset\glstxtrpostlocalunset
2934 \renewcommand*{\glstxtrpostlocalunset}[1]{%
2935   \@glstxtr@entryunitcount@org@localunset{##1}%
2936   \@glst@local@increment@currunitcount{##1}%
2937 }%
2938 \let\@glstxtr@entryunitcount@org@reset\glstxtrpostreset
2939 \renewcommand*{\glstxtrpostreset}[1]{%
2940   \glshasattribute{##1}{unitcount}%
2941   {%
2942     \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
2943     \ifcsundef{\@glstxtr@csname}%
2944       {}%
2945       {\csgdef{\@glstxtr@csname}{0}}%
2946     }%
2947     {}%
2948 }%
2949 \let\@glstxtr@entryunitcount@org@localreset\glstxtrpostlocalreset
2950 \renewcommand*{\glstxtrpostlocalreset}[1]{%
2951   \@glstxtr@entryunitcount@org@localreset{##1}%
2952   \glshasattribute{##1}{unitcount}%
2953   {%
2954     \edef\@glstxtr@csname{\@glstxtr@currentunitcount{##1}}%
2955     \ifcsundef{\@glstxtr@csname}%
2956       {}%
2957       {\csdef{\@glstxtr@csname}{0}}%
2958     }%

```

```

2959    {}%
2960  }%

```

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

```

2961  \let\@cgls@\@cgls@
2962  \let\@cglspl@\@cglspl@

2963  \let\@cGls@\@cGls@
2964  \let\@cGlspl@\@cGlspl@
2965  \let\@cGLS@\@cGLS@
2966  \let\@cGLSpl@\@cGLSpl@

```

Write information to the aux file.

```

2967  \AtEndDocument{\@gls@write@entryunitcounts}%
2968  \renewcommand*{\@gls@entry@unitcount}[3]{%
2969    \csgdef{glo@glstdetoklabel{##1}@prevunit@##3}{##2}%
2970    \ifcsundef{glo@glstdetoklabel{##1}@prevunittotal}%
2971    {\csgdef{glo@glstdetoklabel{##1}@prevunittotal}{##2}}%
2972    {%
2973      \csxdef{glo@glstdetoklabel{##1}@prevunittotal}{
2974        \number\numexpr\csuse{glo@glstdetoklabel{##1}@prevunittotal}+##2}%
2975      }%
2976      \ifcsundef{glo@glstdetoklabel{##1}@prevunitmax}%
2977      {\csgdef{glo@glstdetoklabel{##1}@prevunitmax}{##2}}%
2978      {%
2979        \ifnum\csuse{glo@glstdetoklabel{##1}@prevunitmax}<##2
2980        \csgdef{glo@glstdetoklabel{##1}@prevunitmax}{##2}%
2981        \fi
2982      }%
2983    }%
2984  \let\glsenableentryunitcount\relax
2985  \renewcommand*{\glsenableentrycount}{%
2986    \PackageError{glossaries-extra}{\string\glsenableentrycount\space
2987      can't be used with \string\glsenableentryunitcount}%
2988    {Use one or other but not both commands}%
2989  }%
2990 }
2991 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

2992 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

2993 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
2994   \immediate\write\@auxout
2995   {\string\@gls@entry@unitcount
2996     {\@glsentry}%
2997     {\@glsxtr@currunitcount{\@glsentry}{##1}%
2998     }%

```



```

2999      {#1}}}%
3000 }

```

entryunitcounts

```

3001 \newcommand*{\@gls@write@entryunitcounts}{%
3002   \immediate\write\@auxout
3003     {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}}%
3004   \count@=0\relax
3005   \forallglsentries{\@glsentry}{%
3006     \glshasattribute{\@glsentry}{unitcount}%
3007     {%
3008       \ifglsused{\@glsentry}%
3009       {%
3010         \forlistcsloop
3011           {\@gls@write@entryunitcounts@do}%
3012           {glo@\glsdetoklabel{\@glsentry}@unitlist}%
3013         }%
3014         {}%
3015         \advance\count@ by \@ne
3016       }%
3017     }%
3018   }%
3019   \ifnum\count@=0
3020     \GlossariesExtraWarningNoLine{Entry counting has been enabled
3021       \MessageBreak with \string\glsenableentryunitcount\space but the
3022       \MessageBreak attribute ‘unitcount’ hasn’t
3023       \MessageBreak been assigned to any of the defined
3024       \MessageBreak entries}%
3025   \fi
3026 }

```

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.

```

3027 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

3028   \glsenableentryunitcount

```

Redefine \gls etc:

```

3029   \renewcommand*{\gls}{\cgl}%
3030   \renewcommand*{\Gls}{\cGls}%
3031   \renewcommand*{\glspl}{\cglspl}%
3032   \renewcommand*{\Glspl}{\cGlspl}%
3033   \renewcommand*{\GLS}{\cGLS}%
3034   \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

3035   \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

3036 \let\GlsXtrEnableEntryUnitCounting\@glxtr@setentryunitcountunsetattr
3037 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
3038 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
3039 can't be used with \string\GlsXtrEnableEntryUnitCounting}%
3040 {Use one or other but not both commands}}%
3041 }

```

tcountunsetattr

```

3042 \newcommand*{\@glxtr@setentryunitcountunsetattr}[3]{%
3043 \@for\@glxtr@cat:=#1\do
3044 {%
3045 \ifdefempty{\@glxtr@cat}{}%
3046 {%
3047 \glsssetcategoryattribute{\@glxtr@cat}{entrycount}{#2}%
3048 \glsssetcategoryattribute{\@glxtr@cat}{unitcount}{#3}%
3049 }%
3050 }%
3051 }

```

### 1.3.6 Acronym Modifications

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

nericNewAcronym

```

3052 \renewcommand*{\SetGenericNewAcronym}{%
3053 \let\@Gls@entryname\@Gls@acentryname
3054 \renewcommand{\newacronym}[4][{}]{%
3055 \ifdefempty{\@glsacronymlists}%
3056 {%
3057 \def\@glo@type{\acronymtype}%
3058 \setkeys{glossentry}{##1}%
3059 \DeclareAcronymList{\@glo@type}%
3060 }%
3061 }%
3062 \glskeylisttok{##1}%
3063 \glslabeltok{##2}%
3064 \glsshorttok{##3}%
3065 \glslongtok{##4}%
3066 \newacronymhook
3067 \protected@edef\@do@newglossaryentry{%
3068 \noexpand\newglossaryentry{\the\glslabeltok}%
3069 {%
3070 type=\acronymtype,%

```

```

3071     name={\expandonce{\acronymentry{##2}}},%
3072     sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
3073     text={\the\glsshorttok},%
3074     short={\the\glsshorttok},%
3075     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3076     long={\the\glslongtok},%
3077     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3078     category=acronym,
3079     \GenericAcronymFields,%
3080     \the\glskeylisttok
3081 }%
3082 }%
3083 \@do@newglossaryentry
3084 }%
3085 \renewcommand*{\acrfullfmt}[3]{%
3086   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
3087 \renewcommand*{\Acrfullfmt}[3]{%
3088   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
3089 \renewcommand*{\ACRfullfmt}[3]{%
3090   \glslink[##1]{##2}{%
3091     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
3092 \renewcommand*{\acrfullplfmt}[3]{%
3093   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
3094 \renewcommand*{\Acrfullplfmt}[3]{%
3095   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
3096 \renewcommand*{\ACRfullplfmt}[3]{%
3097   \glslink[##1]{##2}{%
3098     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
3099 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}}%
3100 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}}%
3101 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}}%
3102 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}}%
3103 }

```

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:

```

3104 \let\@glxtr@org@setacronymstyle\setacronymstyle
3105 \let\@glxtr@org@newacronymstyle\newacronymstyle

```

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

```

3106 \newcommand*{\MakeAcronymsAbbreviations}{%
3107   \renewcommand*{\newacronym}[4][{}]{%
3108     \glxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
3109   }%
3110   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
3111   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%

```

```

3112 \renewcommand*{\setacronymstyle}[1]{%
3113     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
3114     unavailable.
3115     Use \string\setabbreviationstyle\space instead.
3116     The original acronym interface can be restored with
3117     \string\RestoreAcronyms}{}%
3118 }%
3119 \renewcommand*{\newacronymstyle}[1]{%
3120     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
3121     available unless you restore the original acronym interface with
3122     \string\RestoreAcronyms}%
3123     \@glxtr@org@newacronymstyle{##1}%
3124 }%
3125 }

```

Switch acronyms to abbreviations:

```

3126 \MakeAcronymsAbbreviations

```

**RestoreAcronyms** Restore acronyms to glossaries interface.

```

3127 \newcommand*{\RestoreAcronyms}{%
3128     \SetGenericNewAcronym
3129     \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
3130     \renewcommand{\acronymfont}[1]{##1}%
3131     \let\setacronymstyle\@glxtr@org@setacronymstyle
3132     \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.

```

3133 \renewcommand*\@gls@link@checkfirsthyper{%
3134     \ifglused{\glslabel}%
3135     {\let\glxtrifwasfirstuse\@secondoftwo}
3136     {\let\glxtrifwasfirstuse\@firstoftwo}%
3137     \@glxtr@org@checkfirsthyper
3138 }
3139 \glssetcategoryattribute{acronym}{regular}{false}%
3140 \setacronymstyle{long-short}%
3141 }

```

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

```

3142 \renewcommand*{\glsacspace}[1]{%
3143     \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
3144     \ifdim\dimen@<\glsacspacemax~\else\space\fi
3145 }

```

**\glsacspacemax** Value used in the above.

```

3146 \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

```
3147 \newcommand*{\@glxtr@reg@glosslist}{{}
```

Save the original definition of `\makeglossaries`:

```
3148 \let\@glxtr@org@makeglossaries\makeglossaries
```

Redefine `\makeglossaries` to take an optional argument. This should be empty for the usual behaviour (all glossaries need processing with an indexing application) or a comma-separated list of glossary labels indicating those glossaries that should be processed with an indexing application.

\makeglossaries

```
3149 \renewcommand*{\makeglossaries}[1] [] {%
3150   \ifblank{#1}%
3151   {\@glxtr@org@makeglossaries}%
3152   {%
3153     \edef\@glxtr@reg@glosslist{#1}%
3154     \ifundef{\glswrite}{\newwrite\glswrite}{}%
3155     \protected@write\@auxout{}{\string\providecommand
3156       \string\@glorder[1]}{}
3157     \protected@write\@auxout{}{\string\providecommand
3158       \string\@istfilename[1]}{}
3159     \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
3160     \protected@write\@auxout{}{\string\@glorder{\glorder}}
3161     \protected@write\@auxout{}{\string\glxtr@makeglossaries{#1}}
3162     \write\@auxout{\string\providecommand\string\@gl@reference[3]}{}}%
```

Iterate through each supplied glossary type and activate it.

```
3163   \@for\@glo@type:=#1\do{%
3164     \ifdefempty{\@glo@type}{}{\@makeglossary{\@glo@type}}%
3165   }%
```

New glossaries must be created before `\makeglossaries`:

```
3166   \renewcommand*\newglossary[4] [] {%
3167     \PackageError{glossaries}{New glossaries
3168       must be created before \string\makeglossaries}{You need
3169       to move \string\makeglossaries\space after all your
3170       \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
3171   \let\@makeglossary\relax
3172   \let\makeglossary\relax
3173   \renewcommand\makeglossaries[1] [] {}%
```

Disable all commands that have no effect after \makeglossaries

```
3174 \let\@disable@onlypremakeg
```

Allow see key:

```
3175 \let\gls@checkseeallowed\relax
```

Adjust \do@seeglossary

```
3176 \renewcommand*\do@seeglossary}[2]{%
3177   \edef\@gls@label{\glsdetoklabel{##1}}%
3178   \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3179   \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3180   {\@glsxtr@org@doseeglossary{##1}{##2}}%
3181   {%
3182     \protected@write\@auxout{}{%
3183       \string\@gls@reference
3184       {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}%
3185     }%
3186   }%
3187 }%
```

Adjust \do@wrglossary

```
3188 \let\@glsxtr@do@wrglossary\do@wrglossary
3189 \def\do@wrglossary{%
3190   \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3191   \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3192   {\@glsxtr@do@wrglossary}%
3193   {\gls@noidxglossary}%
3194 }%
```

Suppress warning about no \makeglossaries

```
3195 \let\warn@nomakeglossaries\relax
3196 \def\warn@noprintglossary{%
3197   \GlossariesWarningNoLine{No \string\printglossary\space
3198     or \string\printglossaries\space
3199     found.^^J(Remove \string\makeglossaries\space if you don't want
3200     any glossaries.)^^JThis document will not have a glossary}%
3201 }%
```

Only warn for glossaries not listed.

```
3202 \renewcommand{\@gls@noref@warn}[1]{%
3203   \edef\@gls@type{##1}%
3204   \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3205   {%
3206     \GlossariesExtraWarning{Can't use
3207       \string\printnoidxglossary[type={\@gls@type}]
3208       when '\@gls@type' is listed in the optional argument of
3209       \string\makeglossaries}%
3210   }%
3211   {%
3212     \GlossariesWarning{Empty glossary for
3213       \string\printnoidxglossary[type={##1}].
```

```

3214      Rerun may be required (or you may have forgotten to use
3215      commands like \string\gls)}%
3216  }%
3217 }%

```

Adjust display number list to check for type:

```

3218 \renewcommand*\glsdisplaynumberlist[1]{%
3219   \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3220   {\@glsxtr@idx@displaynumberlist{##1}}%
3221   {\@glsxtr@noidx@displaynumberlist{##1}}%
3222 }%

```

Adjust entry list:

```

3223 \renewcommand*\glsentrynumberlist[1]{%
3224   \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3225   {\@glsxtr@idx@entrynumberlist{##1}}%
3226   {\@glsxtr@noidx@entrynumberlist{##1}}%
3227 }%

```

Adjust number list loop

```

3228 \renewcommand*\glsnumberlistloop[2]{%
3229   \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3230   {%
3231     \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3232     not available for glossary ‘##1’}{}%
3233   }%
3234   {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
3235 }%

```

Only sanitize sort for normal indexing glossaries.

```

3236 \renewcommand*\glsprestandardsort[3]{%
3237   \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
3238   {%
3239     \glsdosanitizesort
3240   }%
3241   {%
3242     \ifglssanitizesort
3243       \@gls@noidx@sanitizesort
3244     \else
3245       \@gls@noidx@nosanitizesort
3246     \fi
3247   }%
3248 }%

```

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

```

3249 \renewcommand*\new@glossaryentry[2]{%
3250   \PackageError{glossaries-extra}{Glossary entries must be defined
3251   in the preamble\MessageBreak when you use the optional argument
3252   of \string\makeglossaries}{Either move your definitions to the
3253   preamble or don't use the optional argument of

```

```

3254 \string\makeglossaries}%
3255 }%

```

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).

```

3256 \let\@glo@assign@sortkey\@glstr@mixed@assign@sortkey
3257 \renewcommand*\@printgloss@setsort}{%

```

Need to extract just the type value.

```

3258 \expandafter\@glstr@gettype\expandafter,\@glstr@printglossopts,%
3259 type=\glsdefaulttype,\@end@glstr@gettype
3260 \def\@glo@sorttype{\@glo@default@sorttype}%
3261 }%

```

Check automake setting:

```

3262 \ifglautomake
3263 \renewcommand*\@gls@doautomake}{%
3264 \for\@gls@type:=\@glstr@reg@glosslist\do{%
3265 \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
3266 }%
3267 }%
3268 \fi

```

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

```

3269 \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
3270 }%
3271 }

```

The optional argument version of `\makeglossaries` needs an adjustment to `\@printglossary` to allow `\@glo@assign@sortkey` to pick up the glossary type.

`\orgprintglossary` This no longer simply saves `\@printglossary` with `\let` 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.)

```

3272 \newcommand{\@glstr@orgprintglossary}[2]{%
3273 \def\@glo@type{\glsdefaulttype}%

```

Add check here.

```

3274 \def\glossarytitle{%
3275 \ifcsdef{\@glo@type\@glo@type @title}%
3276 {\csuse{\@glo@type\@glo@type @title}}%
3277 {\glossaryname}}%
3278 \def\glossarytoctitle{\glossarytitle}%
3279 \let\org@glossarytitle\glossarytitle
3280 \def\@glossarystyle{%
3281 \ifx\@glossary@default@style\relax
3282 \GlossariesWarning{No default glossary style provided \MessageBreak
3283 for the glossary '\@glo@type'. \MessageBreak
3284 Using deprecated fallback. \MessageBreak
3285 To fix this set the style with \MessageBreak

```



```

3286      \string\setglossarystyle\space or use the \MessageBreak
3287      style key=value option}%
3288  \fi
3289 }%
3290 \def\gls@dotocitle{\gls@settocitle{\@glo@type}}%
3291 \let\org@glossaryentrynumbers\glossaryentrynumbers
3292 \bgroup
3293   \@printgloss@setsort
3294   \setkeys{printgloss}{#1}%
3295   \ifx\glossarytitle\org@glossarytitle
3296   \else
3297     \cslet{\glo@type@\@glo@type @title}{\glossarytitle}%
3298   \fi
3299   \let\currentglossary\@glo@type
3300   \let\org@glossaryentrynumbers\glossaryentrynumbers
3301   \let\glsnonextpages\glsnonextpages
3302   \let\glsnextpages\glsnextpages
3303   \let\nopostdesc\@nopostdesc
3304   \gls@dotocitle
3305   \@glossarystyle
3306   \let\gls@org@glossaryentryfield\glossentry
3307   \let\gls@org@glossarysubentryfield\subglossentry
3308   \renewcommand{\glossentry}[1]{%
3309     \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
3310     \gls@org@glossaryentryfield{##1}%
3311   }%
3312   \renewcommand{\subglossentry}[2]{%
3313     \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
3314     \gls@org@glossarysubentryfield{##1}{##2}%
3315   }%
3316   \@gls@preglossaryhook
3317   #2%
3318 \egroup
3319 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
3320 \global\let\warn@noprintglossary\relax
3321 }

```

\@printglossary Redefine.

```

3322 \renewcommand{\@printglossary}[2]{%
3323   \def\@glsxtr@printglossopts{#1}%
3324   \@glsxtr@orgprintglossary{#1}{#2}%
3325 }

```

Add a key that switches off the entry targets:

```

3326 \define@choicekey{printgloss}{target}[\val\nr]{true,false}[true]{%
3327   \ifcase\nr
3328     \let\@gls@target\glsdohypertarget
3329   \else
3330     \let\@gls@target\@secondoftwo

```

```

3331 \fi
3332 }

```

@makeglossaries For the benefit of makeglossaries

```

3333 \newcommand*{\glxtr@makeglossaries}[1]{%

```

@glxtr@gettype Get just the type.

```

3334 \def\@glxtr@gettype#1,type=#2,#3\@end@glxtr@gettype{%
3335 \def\@glo@type{#2}%
3336 }

```

@assign@sortkey Assign the sort key.

```

3337 \newcommand\@glxtr@mixed@assign@sortkey[1]{%
3338 \edef\@glo@type{\@glo@type}%
3339 \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glxtr@reg@glosslist}%
3340 {%
3341 \@glo@no@assign@sortkey{#1}%
3342 }%
3343 {%
3344 \@glo@assign@sortkey{#1}%
3345 }%
3346 }%

```

Display number list for the regular version:

splaynumberlist

```

3347 \let\@glxtr@idx@displaynumberlist\glsdisplaynumberlist

```

Display number list for the “noidx” version:

splaynumberlist

```

3348 \newcommand*{\@glxtr@noidx@displaynumberlist}[1]{%
3349 \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
3350 \ifdef\@gls@loclist
3351 {%
3352 \def\@gls@noidxloclist@sep{%
3353 \def\@gls@noidxloclist@sep{%
3354 \def\@gls@noidxloclist@sep{%
3355 \glsnumlistsep
3356 }%
3357 \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
3358 }%
3359 }%
3360 \def\@gls@noidxloclist@finalsep{}%
3361 \def\@gls@noidxloclist@prev{}%
3362 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
3363 \@gls@noidxloclist@finalsep
3364 \@gls@noidxloclist@prev
3365 }%

```

```

3366 {%
3367   ??\glsdoifexists{#1}%
3368   {%
3369     \GlossariesWarning{Missing location list for ‘#1’. Either
3370       a rerun is required or you haven’t referenced the entry.}%
3371   }%
3372 }%
3373 }%
3374

```

And for the number list loop:

@numberlistloop

```

3375 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
3376   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
3377   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
3378   \let\@gls@org@glsseeformat\glsseeformat
3379   \let\glsnoidxdisplayloc#2\relax
3380   \let\glsseeformat#3\relax
3381   \ifdef\@gls@loclist
3382   {%
3383     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
3384   }%
3385   {%
3386     ??\glsdoifexists{#1}%
3387     {%
3388       \GlossariesWarning{Missing location list for ‘##1’. Either
3389         a rerun is required or you haven’t referenced the entry.}%
3390     }%
3391   }%
3392   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
3393   \let\glsseeformat\@gls@org@glsseeformat
3394 }%

```

Same for entry number list.

entrynumberlist

```

3395 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
3396   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
3397   \ifdef\@gls@loclist
3398   {%
3399     \glsnoidxloclist{\@gls@loclist}%
3400   }%
3401   {%
3402     ??\glsdoifexists{#1}%
3403     {%
3404       \GlossariesWarning{Missing location list for ‘#1’. Either
3405         a rerun is required or you haven’t referenced the entry.}%
3406     }%
3407   }%
3408 }%

```

entrynumberlist

```
3409 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}
```

x@getgrouptitle Patch.

```
3410 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
3411   \protected@edef\@glsxtr@titlelabel{#1}%
3412   \ifdefvoid\@glsxtr@titlelabel
3413   {%
3414     {%
3415       \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
3416     }%
3417     \ifdefvoid{\@glsxtr@titlelabel}%
3418     {%
3419       \DTLifint{#1}%
3420       {%
3421         \ifnum#1<256\relax
3422           \edef#2{\char#1\relax}%
3423         \else
3424           \edef#2{#1}%
3425         \fi
3426       }%
3427       {%
3428         \ifcsundef{#1groupname}%
3429         {\def#2{#1}}%
3430         {\letcs#2{#1groupname}}%
3431       }%
3432     }%
3433     {%
3434       \let#2\@glsxtr@titlelabel
3435     }%
3436 }
```

g@getgrouptitle Save original definition of \@gls@getgrouptitle

```
3437 \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.

```
3438 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
3439   \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
3440   \@onelevel@sanitize\@glsxtr@titlelabel
3441   \ifcsdef{\@glsxtr@titlelabel}
3442   {\letcs{#2}{\@glsxtr@titlelabel}}%
3443   {\glsxtr@org@getgrouptitle{#1}{#2}}%
3444 }
3445 \let\@gls@getgrouptitle\glsxtrgetgrouptitle
```

trsetgrouptitle Sets the title for the given group label.

```
3446 \newcommand{\glsxtrsetgrouptitle}[2]{%
```

```

3447 \protected@edef\@glstr@titlelabel{glstr@grouptitle@#1}%
3448 \@onelevel@sanitize\@glstr@titlelabel
3449 \csxdef{\@glstr@titlelabel}{#2}%
3450 }

```

`\glsnavigation`   Redefine to use new user-level command.

```

3451 \renewcommand*{\glsnavigation}{%
3452   \def\@gls@between{}%
3453   \ifcsundef{\@gls@hypergroup@list@\@glo@type}%
3454   {%
3455     \def\@gls@list{}%
3456   }%
3457   {%
3458     \expandafter\let\expandafter\@gls@list
3459       \csname \@gls@hypergroup@list@\@glo@type\endcsname
3460   }%
3461   \@for\@gls@tmp:=\@gls@list\do{%
3462     \@gls@between
3463     \glstrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
3464     \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
3465     \let\@gls@between\glshypernavsep
3466   }%
3467 }

```

`@noidx@glossary`

```

3468 \renewcommand*{\@print@noidx@glossary}{%
3469   \ifcsdef{\@glsref@\@glo@type}%
3470   {%
3471     \ifcsdef{\@glo@sortmacro@\@glo@sorttype}%
3472     {%
3473       \csuse{\@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
3474     }%
3475     {%
3476       \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
3477     }%
3478     \glossarysection[\glossarytoctitle]{\glossarytitle}%
3479     \glossarypreamble

```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```

3480   \def\@gls@currentlettergroup{}%
3481   \begin{theglossary}%
3482     \glossaryheader
3483     \glsresetentrylist
3484     \forlistcsloop{\@gls@noidx@do}{\@glsref@\@glo@type}%
3485     \end{theglossary}%
3486     \glossarypostamble
3487   }%
3488   {%

```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```

3489 \glxtrifemptyglossary{\@glo@type}%
3490 {}%
3491 {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
3492 \@gls@noref@warn{\@glo@type}%
3493 }%
3494 }

```

`noidxdisplayloc` Patch to check for range formations.

```

3495 \renewcommand*{\glsnoidxdisplayloc}[4]{%
3496 \setentrycounter[#1]{#2}%
3497 \@glxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
3498 }

```

`xtr@display@loc` Patch to check for range formations.

```

3499 \def\@glxtr@display@loc#1#2\end@glxtr@display@loc#3{%
3500 \ifx#1(\relax
3501 \glxtrdisplaystartloc{#2}{#3}%
3502 \else
3503 \ifx#1)\relax
3504 \glxtrdisplayendloc{#2}{#3}%
3505 \else
3506 \glxtrdisplaysingleloc{#1#2}{#3}%
3507 \fi
3508 \fi
3509 }

```

`isplayingleloc` Single location.

```

3510 \newcommand*{\glxtrdisplaysingleloc}[2]{%
3511 \csuse{#1}{#2}%
3512 }

```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glxtrlocrangefmt`.

`displaystartloc` Start of a location range.

```

3513 \newcommand*{\glxtrdisplaystartloc}[2]{%
3514 \edef\glxtrlocrangefmt{#1}%
3515 \ifx\glxtrlocrangefmt\empty
3516 \def\glxtrlocrangefmt{glsnumberformat}%
3517 \fi
3518 \expandafter\glxtrdisplaysingleloc
3519 \expandafter{\glxtrlocrangefmt}{#2}%
3520 }

```

`trdisplayendloc` End of a location range.

```

3521 \newcommand*{\glxtrdisplayendloc}[2]{%

```

```

3522 \edef\@glsxtr@tmp{#1}%
3523 \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}%
3524 \ifx\glsxtrlocrangefmt\@glsxtr@tmp
3525 \else
3526   \GlossariesExtraWarning{Mismatched end location range
3527     (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%
3528 \fi
3529 \expandafter\glsxtrdisplayendlochook\expandafter{\@glsxtr@tmp}{#2}%
3530 \expandafter\glsxtrdisplaysingleloc
3531   \expandafter{\glsxtrlocrangefmt}{#2}%
3532 \def\glsxtrlocrangefmt{}%
3533 }

```

`splayendlochook` Allow the user to hook into the end of range command.

```

3534 \newcommand*{\glsxtrdisplayendlochook}[2]{}

```

`sxtrlocrangefmt` Current range format. Empty if not in a range.

```

3535 \newcommand*{\glsxtrlocrangefmt}{}

```

`ls@removespaces` Redefine to allow adjustments to location hyperlink.

```

3536 \def\@gls@removespaces#1 #2\@nil{%
3537   \toks@=\expandafter{\the\toks@#1}%
3538   \ifx\#2\%
3539     \edef\x{\the\toks@}%
3540     \ifx\x\empty
3541       \else
3542         \glsxtrlocationhyperlink{\glsentrycounter}{\@glo@counterprefix}{\the\toks@}%
3543       \fi
3544     \else
3545       \@gls@ReturnAfterFi{%
3546         \@gls@removespaces#2\@nil
3547       }%
3548     \fi
3549 }

```

`cationhyperlink`

```

3550 \newcommand*{\glsxtrlocationhyperlink}[3]{%
3551   \ifdefvoid\glsxtrsupplocationurl
3552     {%
3553       \hyperlink{#1#2#3}{#3}%
3554     }%
3555     {%
3556       \hyperref{\glsxtrsupplocationurl}{#1#2#3}{#3}%
3557     }%
3558 }

```

`supphypernumber`

```

3559 \newcommand*{\glsxtrsupphypernumber}[1]{%
3560   {%

```

```

3561 \glshasattribute{\glscurrententrylabel}{externallocation}%
3562 {%
3563   \def\glsxtrsupplocationurl{%
3564     \glsggetattribute{\glscurrententrylabel}{externallocation}}%
3565   }%
3566   {%
3567     \def\glsxtrsupplocationurl{%
3568       }%
3569   \glshypernumber{#1}%
3570 }%
3571 }

```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```

3572 \renewcommand{\@print@glossary}{%
3573   \makeatletter
3574   \@input{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
3575   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
3576   {}%
3577   {\glsxtrNoGlossaryWarning{\@glo@type}}%
3578   \ifglsxindy
3579     \ifcsundef{@xdy@\@glo@type @language}%
3580     {%
3581       \edef\@do@auxoutstuff{%
3582         \noexpand\AtEndDocument{%
3583           \noexpand\immediate\noexpand\write\@auxout{%
3584             \string\providecommand\string\@xdylanguage[2]{}}%
3585           \noexpand\immediate\noexpand\write\@auxout{%
3586             \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
3587         }%
3588       }%
3589     }%
3590     {%
3591       \edef\@do@auxoutstuff{%
3592         \noexpand\AtEndDocument{%
3593           \noexpand\immediate\noexpand\write\@auxout{%
3594             \string\providecommand\string\@xdylanguage[2]{}}%
3595           \noexpand\immediate\noexpand\write\@auxout{%
3596             \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
3597               @language\endcsname}}%
3598         }%
3599       }%
3600     }%
3601   \@do@auxoutstuff
3602   \edef\@do@auxoutstuff{%
3603     \noexpand\AtEndDocument{%
3604       \noexpand\immediate\noexpand\write\@auxout{%
3605         \string\providecommand\string\@gls@codepage[2]{}}%

```



```

3606         \noexpand\immediate\noexpand\write\@auxout{%
3607         \string\@gls@codepage{\@glo@type}{\@gls@codepage}}}%
3608     }%
3609 }%
3610 \do@auxoutstuff
3611 \fi
3612 \renewcommand*{\@warn@nomakeglossaries}{%
3613     \GlossariesWarningNoLine{\string\makeglossaries\space
3614     hasn't been used,^^Jthe glossaries will not be updated}}%
3615 }%
3616 }

```

Setup the warning text to display if the external file for the given glossary is missing.

`\GlsWarningHead` Header message.

```

3617 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
3618     This document is incomplete. The external file associated with
3619     the glossary '#1' (which should be called \texttt{#2})
3620     hasn't been created.%
3621 }

```

`\GlsWarningEmptyStart` No entries have been added to the glossary.

```

3622 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
3623     This has probably happened because there are no entries defined
3624     in this glossary.%
3625 }

```

`\GlsWarningEmptyMain` The default “main” glossary is empty.

```

3626 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
3627     If you don't want this glossary,
3628     add \texttt{nomain} to your package option list when you load
3629     \texttt{glossaries-extra.sty}. For example:%
3630 }

```

`\GlsWarningEmptyNotMain` A glossary that isn't the default “main” glossary is empty.

```

3631 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
3632     Did you forget to use \texttt{type=#1} when you defined your
3633     entries? If you tried to load entries into this glossary with
3634     \texttt{\string\loadglsentries} did you remember to use
3635     \texttt{[#1]} as the optional argument? If you did, check that
3636     the definitions in the file you loaded all had the type set
3637     to \texttt{\string\glsdefaulttype}.%
3638 }

```

`\GlsWarningCheckFile` Advisory message to check the file contents.

```

3639 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
3640     Check the contents of the file \texttt{#1}. If
3641     it's empty, that means you haven't indexed any of your entries in this
3642     glossary (using commands like \texttt{\string\gls} or

```

```

3643 \texttt{\string\glsadd}) so this list can't be generated.
3644 If the file isn't empty, the document build process hasn't been
3645 completed.%
3646 }

```

WarningAutoMake Message when automake option has been used.

```

3647 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
3648   You may need to rerun \LaTeX. If you already have, it may be that
3649   \TeX's shell escape doesn't allow you to run
3650   \ifglxindy xindy\else makeindex\fi. Check the
3651   transcript file \texttt{\jobname.log}. If the shell escape is
3652   disabled, try one of the following:
3653
3654   \begin{itemize}
3655     \item Run the external (Lua) application:
3656
3657       \texttt{makeglossaries-lite.lua \string"\jobname\string"}
3658
3659     \item Run the external (Perl) application:
3660
3661       \texttt{makeglossaries \string"\jobname\string"}
3662   \end{itemize}
3663
3664   Then rerun \LaTeX\ on this document.
3665   \GlossariesExtraWarning{Rerun required to build the
3666   glossary '#1' or check TeX's shell escape allows
3667   you to run \ifglxindy xindy\else makeindex\fi}%
3668 }

```

WarningMismatch Mismatching \makenoidxglossaries.

```

3669 \newcommand{\GlsXtrNoGlsWarningMismatch}{%
3670   You need to either replace \texttt{\string\makenoidxglossaries}
3671   with \texttt{\string\makeglossaries} or replace
3672   \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
3673   \texttt{\string\printnoidxglossary}
3674   (or \texttt{\string\printnoidxglossaries}) and then rebuild
3675   this document.%
3676 }

```

WarningBuildInfo Build advice.

```

3677 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
3678   Try one of the following:
3679   \begin{itemize}
3680     \item Add \texttt{automake} to your package option list when you load
3681       \texttt{glossaries-extra.sty}. For example:
3682
3683       \texttt{\string\usepackage[automake]%
3684         \glsopenbrace glossaries-extra\glsclosebrace}

```

```

3685
3686 \item Run the external (Lua) application:
3687
3688 \texttt{makeglossaries-lite.lua \string"\jobname\string"}
3689
3690 \item Run the external (Perl) application:
3691
3692 \texttt{makeglossaries \string"\jobname\string"}
3693 \end{itemize}
3694
3695 Then rerun \LaTeX\ on this document.%
3696 }

```

oGlsWarningTail Final paragraph.

```

3697 \newcommand{\GlsXtrNoGlsWarningTail}{%
3698 This message will be removed once the problem has been fixed.%
3699 }

```

GlsWarningNoOut No out file created. Build advice.

```

3700 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
3701 The file \texttt{#1} doesn't exist. This most likely means you haven't used
3702 \texttt{\string\makeglossaries} or you have used
3703 \texttt{\string\nofiles}. If this is just a draft version of the
3704 document, you can suppress this message using the
3705 \texttt{\nomissingglstext} package option.%
3706 }

```

glossarywarning

```

3707 \newcommand*{@@glsxtr@defaultnoglossarywarning}[1]{%
3708 \glossarysection[\glossarytoctitle]{\glossarytitle}
3709 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glo@type @in\endcsname}
3710 \par
3711 \glsxtrifemptyglossary{#1}%
3712 {%
3713 \GlsXtrNoGlsWarningEmptyStart\space
3714 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
3715 \medskip
3716 \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
3717 \glsopenbrace glossaries-extra\glsclosebrace}
3718 \medskip
3719 }%
3720 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
3721 }%
3722 {%
3723 \IfFileExists{\jobname.\csname @glo@type @out\endcsname}
3724 {%
3725 \GlsXtrNoGlsWarningCheckFile
3726 {\jobname.\csname @glo@type @out\endcsname}
3727 }

```

```

3728 \ifglsautomake
3729
3730 \GlsXtrNoGlsWarningAutoMake{#1}
3731
3732 \else
3733
3734 \ifthenelse{\equal{#1}{main}}{%
3735 {%
3736 \GlsXtrNoGlsWarningEmptyMain\par
3737 \medskip
3738 \noindent\texttt{\string\usepackage[nomain]%
3739 \glsopenbrace glossaries-extra\glsclosebrace}
3740 \medskip
3741 }%
3742 }%
3743
3744 \ifdefequal\makeglossaries\@no@makeglossaries
3745 {%
3746 \GlsXtrNoGlsWarningMisMatch
3747 }%
3748 {%
3749 \GlsXtrNoGlsWarningBuildInfo
3750 }%
3751 \fi
3752 }%
3753 {%
3754 \GlsXtrNoGlsWarningNoOut
3755 {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
3756 }%
3757 }%
3758 \par
3759 \GlsXtrNoGlsWarningTail
3760 }

```

Provide some commands to accompany the record option for use with **bib2gls**.

**xtresourcefile** Since it's dangerous for an external application to create a file with a .tex extension, as from v1.11 this enforces a .glstex extension to avoid conflict.

```

3761 \newcommand*\glstxtresourcefile[2] [] {%
3762 \glstxt@writefields
3763 \protected@write\@auxout{}\string\glstxt@resource{#1}{#2}}%
3764 \let\@glstxt@org@see@noindex\@gls@see@noindex
3765 \let\@gls@see@noindex\relax
3766 \IfFileExists{#2.glstex}%
3767 {%

```

Can't scope \input so save and restore the category code of @ to allow for internal commands in the location list.

```

3768 \edef\@bibgls@restorat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
3769 \makeatletter

```

```

3770 \input{#2.glstex}%
3771 \bibgls@restoreat
3772 }%
3773 {%
3774 \GlossariesExtraWarning{No file ‘#2.glstex’}%
3775 }%
3776 \let\@gls@see@noindex\@glsxtr@org@see@noindex
3777 }
3778 \@onlypreamble\glsxtrresourcefile

trresourcecount
3779 \newcount\glsxtrresourcecount

trLoadResources Short cut that uses \glsxtrresourcefile with \jobname as the mandatory argument.
3780 \newcommand*{\GlsXtrLoadResources}[1][{}]{%
3781 \ifnum\glsxtrresourcecount=0\relax
3782 \glsxtrresourcefile[#1]{\jobname}%
3783 \else
3784 \glsxtrresourcefile[#1]{\jobname-\the\glsxtrresourcecount}%
3785 \fi
3786 \advance\glsxtrresourcecount by 1\relax
3787 }

glsxtr@resource
3788 \newcommand*{\glsxtr@resource}[2]{%

\glsxtr@fields
3789 \newcommand*{\glsxtr@fields}[1]{%

xtr@texencoding
3790 \newcommand*{\glsxtr@texencoding}[1]{%

\glsxtr@langtag
3791 \newcommand*{\glsxtr@langtag}[1]{%

@pluralsuffixes
3792 \newcommand*{\glsxtr@pluralsuffixes}[4]{%

tr@shortcutsval
3793 \newcommand*{\glsxtr@shortcutsval}[1]{%

sxtr@linkprefix
3794 \newcommand*{\glsxtr@linkprefix}[1]{%

xtr@writefields This information only needs to be written once, so disable it after it's been used.
3795 \newcommand*{\glsxtr@writefields}{%

```

```

3796 \protected@write\@auxout{}%
3797   {\string\providecommand*{\string\glstr@fields}[1]{}}%
3798 \protected@write\@auxout{}%
3799   {\string\providecommand*{\string\glstr@resource}[2]{}}%
3800 \protected@write\@auxout{}%
3801   {\string\providecommand*{\string\glstr@pluralsuffixes}[4]{}}%
3802 \protected@write\@auxout{}%
3803   {\string\providecommand*{\string\glstr@shortcutsval}[1]{}}%
3804 \protected@write\@auxout{}%
3805   {\string\providecommand*{\string\glstr@linkprefix}[1]{}}%
3806 \protected@write\@auxout{}{\string\glstr@fields{\@glstr@keymap}}%

```

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 `\glstrresourcefile`.

```

3807 \ifdef\CurrentTrackedLanguageTag
3808 {%
3809   \protected@write\@auxout{}{%
3810     \string\glstr@langtag{\CurrentTrackedLanguageTag}}%
3811   }%
3812   {}%
3813   \protected@write\@auxout{}{\string\glstr@pluralsuffixes
3814     {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
3815     {\glstrabbrvpluralsuffix}}%
3816 \ifdef\inputencodingname
3817 {%
3818   \protected@write\@auxout{}{\string\glstr@texencoding{\inputencodingname}}%
3819   }%
3820   {}%

```

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.)

```

3821   \ifpackageloaded{fontspec}%
3822   {\protected@write\@auxout{}{\string\glstr@texencoding{utf8}}}%
3823   {}%
3824   }%
3825 \protected@write\@auxout{}{\string\glstr@shortcutsval{\@glstr@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.

```

3826 \AtBeginDocument
3827   {\protected@write\@auxout{}{\string\glstr@linkprefix{\glolinkprefix}}}%
3828 \let\glstr@writefields\relax

```

If the `automake` option is on, try running `bib2gls` if the aux file exists.

```

3829 \ifglstrautomake
3830   \IfFileExists{\jobname.aux}%
3831   {\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`.

```

3832 \ifx\@gls@doautomake\@gls@doautomake@err
3833 \let\@gls@doautomake\relax
3834 \fi
3835 \fi
3836 }

```

do@automake@err

```

3837 \newcommand*{\@gls@doautomake@err}{%
3838 \PackageError{glossaries}{You must use
3839 \string\makeglossaries\space with automake=true}
3840 {%
3841 Either remove the automake=true setting or
3842 add \string\makeglossaries\space to your document preamble.%
3843 }%
3844 }

```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

```

3845 \newcommand*{\glsxtr@record}[5]{}

```

r@counterrecord Aux file command.

```

3846 \newcommand*{\glsxtr@counterrecord}[3]{%
3847 \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
3848 }

```

unterrecordhook Hook used by \@glsxtr@dorecord.

```

3849 \newcommand*{\@glsxtr@counterrecordhook}{}

```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```

3850 \newcommand*{\GlsXtrRecordCounter}[1]{%
3851 \@glsxtr@recordcounter{#1}%
3852 }
3853 \@onlypreamble\GlsXtrRecordCounter

```

docounterrecord

```

3854 \newcommand*{\@glsxtr@docounterrecord}[1]{%
3855 \protected@write\@auxout{}{\string\glsxtr@counterrecord
3856 {\@gls@label}{#1}{\csuse{the#1}}}%
3857 }

```

ntunsrtglossary Similar to \printnoidxglossary but it displays all entries defined for the given glossary without sorting.

```

3858 \newcommand*{\printunsrtglossary}{%
3859 \@ifstar\s@printunsrtglossary\@printunsrtglossary
3860 }

```

ntunsrtglossary Unstarred version.

```
3861 \newcommand*{\@printunsrtglossary}[1][]{%
3862   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
3863 }
```

ntunsrtglossary Starred version.

```
3864 \newcommand*{\s@printunsrtglossary}[2][]{%
3865   \begin{group}
3866     #2%
3867     \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
3868   \end{group}
3869 }
```

unsrtglossaries Similar to \printnoidxglossaries but it displays all entries defined for the given glossary without sorting.

```
3870 \newcommand*{\printunsrtglossaries}{%
3871   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
3872 }
```

@unsrt@glossary

```
3873 \newcommand*{\@print@unsrt@glossary}{%
3874   \glossarysection[\glossarytoctitle]{\glossarytitle}%
3875   \glossarypreamble
3876   check for empty list
3877   \glstrifemptyglossary{\@glo@type}%
3878   {%
3879     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
3880   }%
3881   \key@ifundefined{glossentry}{group}%
3882   {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
3883   {\let\@gls@getgrouptitle\@glstr@unsrt@getgrouptitle}%
3884   \def\@gls@currentlettergroup{}}%
```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```
3885 \def\@glstr@doglossary{%
3886   \begin{theglossary}%
3887   \glossaryheader
3888   \glsresetentrylist
3889 }%
3890 \expandafter\@for\expandafter\glscurrententrylabel\expandafter
3891   :\expandafter=\csname glolist@\@glo@type\endcsname\do{%
3892   \ifdefempty{\glscurrententrylabel}
3893   }%
3894   {\eappto\@glstr@doglossary{%
3895     \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}}%
3896 }
```



```

3897 \appto\@glstr@doglossary{\end{theglossary}}%
3898 \glstr@doglossary
3899 }%
3900 \glossarypostamble
3901 }

```

lossary@handler

```

3902 \newcommand{\@printunsrt@glossary@handler}[1]{%
3903 \xdef\glscurrententrylabel{#1}%
3904 \printunsrtglossaryhandler\glscurrententrylabel
3905 }

```

glossaryhandler

```

3906 \newcommand{\printunsrtglossaryhandler}[1]{%
3907 \glstrunsrtdo{#1}%
3908 }

```

srtglossaryunit

```

3909 \newcommand{\print@op@unsrtglossaryunit}[2][ ]{%
3910 \s@printunsrtglossary[type=\gldefaulttype,#1]{%
3911 \printunsrtglossaryunitsetup{#2}%
3912 }%
3913 }

```

ossaryunitsetup

```

3914 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
3915 \renewcommand{\printunsrtglossaryhandler}[1]{%
3916 \glstrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
3917 {\glstrunsrtdo{##1}}%
3918 }%
3919 }%
3920 \ifcsundef{theH#1}%
3921 {%
3922 \renewcommand*{\glolinkprefix}{record.#1.\csuse{the#1}.}%
3923 }%
3924 {%
3925 \renewcommand*{\glolinkprefix}{record.#1.\csuse{theH#1}.}%
3926 }%
3927 \renewcommand*{\glossarysection}[2][ ]{%
3928 \appto\glossarypostamble{\glspare\medskip\glspare}%
3929 }

```

srtglossaryunit

```

3930 \newcommand{\print@noop@unsrtglossaryunit}[2][ ]{%
3931 \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
3932 requires the record=only or record=alsoindex package option}{}%
3933 }

```

t@getgrouptitle

```
3934 \newrobustcmd*{\@glsxtr@unsrt@getgrouptitle}[2]{%
3935   \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
3936   \@onelevel@sanitize\@glsxtr@titlelabel
3937   \ifcsdef{\@glsxtr@titlelabel}
3938     {\letcs{#2}{\@glsxtr@titlelabel}}%
3939     {\def#2{#1}}%
3940 }
```

\glsxtrunsrtdo Provide a user-level call to \@glsxtr@noidx@do to make it easier to define a new handler.

```
3941 \newcommand{\glsxtrunsrtdo}{\@glsxtr@noidx@do}
```

glsxtr@noidx@do Minor modification of \@gls@noidx@do to check for location field if present.

```
3942 \newcommand{\@glsxtr@noidx@do}[1]{%
3943   \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
3944   \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@location}%
3945   \ifglshasparent{#1}%
3946   {%
3947     \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
3948     \ifdefvoid{\@gls@location}%
3949     {%
3950       \ifdefvoid{\@gls@loclist}%
3951       {%
3952         \subglossentry{\@gls@level}{#1}{}%
3953       }%
3954       {%
3955         \subglossentry{\@gls@level}{#1}%
3956         {%
3957           \glossaryentrynumbers{\@glsnoidxloclist{\@gls@loclist}}%
3958         }%
3959       }%
3960     }%
3961     {%
3962       \subglossentry{\@gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
3963     }%
3964   }%
3965   {%
3966     \key@ifundefined{glossentry}{group}%
3967     {%
3968       \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
3969       \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
3970     }%
3971     {%
3972       \protected@xdef\@glo@thislettergrp{%
3973         \csuse{glo@\glsdetoklabel{#1}@group}}%
3974     }%
3975     \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
3976     {}%
```

```

3977 {%
3978   \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
3979   \expandafter\gls@groupheading\expandafter
3980   {\@glo@thislettergrp}%
3981 }%
3982 \global\let\@gls@currentlettergroup\@glo@thislettergrp
3983 \ifdefvoid{\@gls@location}%
3984 {%
3985   \ifdefvoid{\@gls@loclist}
3986   {%
3987     \glossentry{#1}{}%
3988   }%
3989   {%
3990     \glossentry{#1}%
3991     {%
3992       \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
3993     }%
3994   }%
3995 }%
3996 {%
3997   \glossentry{#1}%
3998   {%
3999     \glossaryentrynumbers{\@gls@location}%
4000   }%
4001 }%
4002 }%
4003 }

```

## 1.4 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.

```

4004 \ifpackageloaded{glossaries-accsupp}
4005 {

```

Define (or redefine) commands to use the accessibility information.

`\glsaccessname` Display the name value (no link and no check for existence).

```

4006   \newcommand*{\glsaccessname}[1]{%
4007     \glsnameaccessdisplay
4008     {%
4009       \glsentryname{#1}%
4010     }%
4011     {#1}%
4012   }

```

`\Glsaccessname` Display the name value (no link and no check for existence) with the first letter converted to upper case.

```

4013 \newcommand*{\Glsaccessname}[1]{%
4014   \glsnameaccessdisplay
4015   {%
4016     \Glsentryname{#1}%
4017   }%
4018   {#1}%
4019 }
```

`\GLSaccessname` Display the name value (no link and no check for existence) converted to upper case.

```

4020 \newcommand*{\GLSaccessname}[1]{%
4021   \glsnameaccessdisplay
4022   {%
4023     \mfirstucMakeUppercase{\glsentryname{#1}}%
4024   }%
4025   {#1}%
4026 }
```

`\glsaccesstext` Display the text value (no link and no check for existence).

```

4027 \newcommand*{\glsaccesstext}[1]{%
4028   \glstextaccessdisplay
4029   {%
4030     \glsentrytext{#1}%
4031   }%
4032   {#1}%
4033 }
```

`\Glsaccesstext` Display the text value (no link and no check for existence) with the first letter converted to upper case.

```

4034 \newcommand*{\Glsaccesstext}[1]{%
4035   \glstextaccessdisplay
4036   {%
4037     \Glsentrytext{#1}%
4038   }%
4039   {#1}%
4040 }
```

`\GLSaccesstext` Display the text value (no link and no check for existence) converted to upper case.

```

4041 \newcommand*{\GLSaccesstext}[1]{%
4042   \glstextaccessdisplay
4043   {%
4044     \mfirstucMakeUppercase{\glsentrytext{#1}}%
4045   }%
4046   {#1}%
4047 }
```

`glsaccessplural` Display the plural value (no link and no check for existence).

```

4048 \newcommand*\glsaccessplural}[1]{%
4049   \glspluralaccessdisplay
4050   {%
4051     \glsentryplural{#1}%
4052   }%
4053   {#1}%
4054 }

```

**Glsaccessplural** Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```

4055 \newcommand*\Glsaccessplural}[1]{%
4056   \glspluralaccessdisplay
4057   {%
4058     \Glsentryplural{#1}%
4059   }%
4060   {#1}%
4061 }

```

**GLSaccessplural** Display the plural value (no link and no check for existence) converted to upper case.

```

4062 \newcommand*\GLSaccessplural}[1]{%
4063   \glspluralaccessdisplay
4064   {%
4065     \mfirstucMakeUppercase{\glsentryplural{#1}}%
4066   }%
4067   {#1}%
4068 }

```

**\glsaccessfirst** Display the first value (no link and no check for existence).

```

4069 \newcommand*\glsaccessfirst}[1]{%
4070   \glsfirstaccessdisplay
4071   {%
4072     \glsentryfirst{#1}%
4073   }%
4074   {#1}%
4075 }

```

**\Glsaccessfirst** Display the first value (no link and no check for existence) with the first letter converted to upper case.

```

4076 \newcommand*\Glsaccessfirst}[1]{%
4077   \glsfirstaccessdisplay
4078   {%
4079     \Glsentryfirst{#1}%
4080   }%
4081   {#1}%
4082 }

```

**\GLSaccessfirst** Display the first value (no link and no check for existence) converted to upper case.

```

4083 \newcommand*\GLSaccessfirst}[1]{%

```

```

4084 \glsfirstaccessdisplay
4085 {%
4086 \mfirstucMakeUppercase{\glstentryfirst{#1}}%
4087 }%
4088 {#1}%
4089 }

```

`\glsfirstplural` Display the firstplural value (no link and no check for existence).

```

4090 \newcommand*{\glsaccessfirstplural}[1]{%
4091 \glsfirstpluralaccessdisplay
4092 {%
4093 \glstentryfirstplural{#1}%
4094 }%
4095 {#1}%
4096 }

```

`\Glsfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

4097 \newcommand*{\Glsaccessfirstplural}[1]{%
4098 \glsfirstpluralaccessdisplay
4099 {%
4100 \Glstentryfirstplural{#1}%
4101 }%
4102 {#1}%
4103 }

```

`\glsfirstplural` Display the firstplural value (no link and no check for existence) converted to upper case.

```

4104 \newcommand*{\GLSaccessfirstplural}[1]{%
4105 \glsfirstpluralaccessdisplay
4106 {%
4107 \mfirstucMakeUppercase{\glstentryfirstplural{#1}}%
4108 }%
4109 {#1}%
4110 }

```

`\glsaccesssymbol` Display the symbol value (no link and no check for existence).

```

4111 \newcommand*{\glsaccesssymbol}[1]{%
4112 \glssymbolaccessdisplay
4113 {%
4114 \glstentrysymbol{#1}%
4115 }%
4116 {#1}%
4117 }

```

`\Glsaccesssymbol` Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

4118 \newcommand*{\GLSaccesssymbol}[1]{%
4119 \glssymbolaccessdisplay

```

```

4120   {%
4121       \Glsentrysymbol{#1}%
4122   }%
4123   {#1}%
4124 }

```

**GLSaccesssymbol** Display the symbol value (no link and no check for existence) converted to upper case.

```

4125 \newcommand*{\GLSaccesssymbol}[1]{%
4126     \glssymbolaccessdisplay
4127     {%
4128         \mfirstucMakeUppercase{\glentrysymbol{#1}}%
4129     }%
4130     {#1}%
4131 }

```

**esssymbolplural** Display the symbolplural value (no link and no check for existence).

```

4132 \newcommand*{\glaccesssymbolplural}[1]{%
4133     \glssymbolpluralaccessdisplay
4134     {%
4135         \glentrysymbolplural{#1}%
4136     }%
4137     {#1}%
4138 }

```

**esssymbolplural** Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

4139 \newcommand*{\Glsaccesssymbolplural}[1]{%
4140     \glssymbolpluralaccessdisplay
4141     {%
4142         \Glsentrysymbolplural{#1}%
4143     }%
4144     {#1}%
4145 }

```

**esssymbolplural** Display the symbolplural value (no link and no check for existence) converted to upper case.

```

4146 \newcommand*{\GLSaccesssymbolplural}[1]{%
4147     \glssymbolpluralaccessdisplay
4148     {%
4149         \mfirstucMakeUppercase{\glentrysymbolplural{#1}}%
4150     }%
4151     {#1}%
4152 }

```

**\glsaccessdesc** Display the desc value (no link and no check for existence).

```

4153 \newcommand*{\glsaccessdesc}[1]{%
4154     \glsdescriptionaccessdisplay
4155     {%
4156         \glentrydesc{#1}%

```

```

4157 }%
4158 {#1}%
4159 }

```

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```

4160 \newcommand*{\Glsaccessdesc}[1]{%
4161   \glsdescriptionaccessdisplay
4162   {%
4163     \Glsentrydesc{#1}%
4164   }%
4165   {#1}%
4166 }

```

`\GLSaccessdesc` Display the desc value (no link and no check for existence) converted to upper case.

```

4167 \newcommand*{\GLSaccessdesc}[1]{%
4168   \glsdescriptionaccessdisplay
4169   {%
4170     \mfirstucMakeUppercase{\glsentrydesc{#1}}%
4171   }%
4172   {#1}%
4173 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence).

```

4174 \newcommand*{\glsaccessdescplural}[1]{%
4175   \glsdescriptionpluralaccessdisplay
4176   {%
4177     \glsentrydescplural{#1}%
4178   }%
4179   {#1}%
4180 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```

4181 \newcommand*{\Glsaccessdescplural}[1]{%
4182   \glsdescriptionpluralaccessdisplay
4183   {%
4184     \Glsentrydescplural{#1}%
4185   }%
4186   {#1}%
4187 }

```

`ccessdescplural` Display the descplural value (no link and no check for existence) converted to upper case.

```

4188 \newcommand*{\GLSaccessdescplural}[1]{%
4189   \glsdescriptionpluralaccessdisplay
4190   {%
4191     \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
4192   }%

```



```

4193     {#1}%
4194 }

```

`\glsaccessshort` Display the short form (no link and no check for existence).

```

4195 \newcommand*\glsaccessshort}[1]{%
4196   \glsshortaccessdisplay
4197   {%
4198     \glentryshort{#1}%
4199   }%
4200   {#1}%
4201 }

```

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

4202 \newcommand*\Glsaccessshort}[1]{%
4203   \glsshortaccessdisplay
4204   {%
4205     \Glsentryshort{#1}%
4206   }%
4207   {#1}%
4208 }

```

`\GLSaccessshort` Display the short value (no link and no check for existence) converted to upper case.

```

4209 \newcommand*\GLSaccessshort}[1]{%
4210   \glsshortaccessdisplay
4211   {%
4212     \mfirstucMakeUppercase{\glentryshort{#1}}%
4213   }%
4214   {#1}%
4215 }

```

`lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

4216 \newcommand*\glsaccessshortpl}[1]{%
4217   \glsshortpluralaccessdisplay
4218   {%
4219     \glentryshortpl{#1}%
4220   }%
4221   {#1}%
4222 }

```

`lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

4223 \newcommand*\Glsaccessshortpl}[1]{%
4224   \glsshortpluralaccessdisplay
4225   {%
4226     \Glsentryshortpl{#1}%
4227   }%
4228   {#1}%
4229 }

```

**LSaccessshortpl** Display the shortplural value (no link and no check for existence) converted to upper case.

```
4230 \newcommand*{\LSaccessshortpl}[1]{%
4231   \glsshortpluralaccessdisplay
4232   {%
4233     \mfirstucMakeUppercase{\glsentryshortpl{#1}}%
4234   }%
4235   {#1}%
4236 }
```

**\glsaccesslong** Display the long form (no link and no check for existence).

```
4237 \newcommand*{\glsaccesslong}[1]{%
4238   \glslongaccessdisplay{\glsentrylong{#1}}{#1}%
4239 }
```

**\Glsaccesslong** Display the long form (no link and no check for existence).

```
4240
4241 \newcommand*{\Glsaccesslong}[1]{%
4242   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
4243 }
```

**\GLSaccesslong** Display the long value (no link and no check for existence) converted to upper case.

```
4244 \newcommand*{\GLSaccesslong}[1]{%
4245   \glslongaccessdisplay
4246   {%
4247     \mfirstucMakeUppercase{\glsentrylong{#1}}%
4248   }%
4249   {#1}%
4250 }
```

**glsaccesslongpl** Display the long plural form (no link and no check for existence).

```
4251 \newcommand*{\glsaccesslongpl}[1]{%
4252   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
4253 }
```

**Glsaccesslongpl** Display the long plural form (no link and no check for existence).

```
4254
4255 \newcommand*{\Glsaccesslongpl}[1]{%
4256   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
4257 }
```

**GLSaccesslongpl** Display the longplural value (no link and no check for existence) converted to upper case.

```
4258 \newcommand*{\GLSaccesslongpl}[1]{%
4259   \glslongpluralaccessdisplay
4260   {%
4261     \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
4262   }%
4263   {#1}%
4264 }
```

End of if part

4265 }

4266 {

No accessibility support. Just define these commands to do `\glentry<xxx>`

`\glsaccessname` Display the name value (no link and no check for existence).

4267 `\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.

4268 `\newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}`

`\GLSaccessname` Display the name value (no link and no check for existence). converted to upper case.

4269 `\newcommand*{\GLSaccessname}[1]{%`

4270 `\protect\mfirstucMakeUppercase{\glentryname{#1}}}`

`\glsaccesstext` Display the text value (no link and no check for existence).

4271 `\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.

4272 `\newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}`

`\GLSaccesstext` Display the text value (no link and no check for existence). converted to upper case.

4273 `\newcommand*{\GLSaccesstext}[1]{%`

4274 `\protect\mfirstucMakeUppercase{\glentrytext{#1}}}`

`glsaccessplural` Display the plural value (no link and no check for existence).

4275 `\newcommand*{\glsaccessplural}[1]{\glentryplural{#1}}`

`Glsaccessplural` Display the plural value (no link and no check for existence) with the first letter converted to upper case.

4276 `\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}`

`GLSaccessplural` Display the plural value (no link and no check for existence). converted to upper case.

4277 `\newcommand*{\GLSaccessplural}[1]{%`

4278 `\protect\mfirstucMakeUppercase{\glentryplural{#1}}}`

`\glsaccessfirst` Display the first value (no link and no check for existence).

4279 `\newcommand*{\glsaccessfirst}[1]{\glentryfirst{#1}}`

`\Glsaccessfirst` Display the first value (no link and no check for existence) with the first letter converted to upper case.

4280 `\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}`

`\GLSaccessfirst` Display the first value (no link and no check for existence). converted to upper case.

```

4281 \newcommand*{\GLSaccessfirst}[1]{%
4282 \protect\mfirstucMakeUppercase{\glentryfirst{#1}}}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```

4283 \newcommand*{\glsaccessfirstplural}[1]{\glentryfirstplural{#1}}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

4284 \newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.

```

4285 \newcommand*{\GLSaccessfirstplural}[1]{%
4286 \protect\mfirstucMakeUppercase{\glentryfirstplural{#1}}}
```

`glsaccesssymbol` Display the symbol value (no link and no check for existence).

```

4287 \newcommand*{\glsaccesssymbol}[1]{\glentrysymbol{#1}}
```

`Glsaccesssymbol` Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

4288 \newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}
```

`GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.

```

4289 \newcommand*{\GLSaccesssymbol}[1]{%
4290 \protect\mfirstucMakeUppercase{\glentrysymbol{#1}}}
```

`esssymbolplural` Display the symbolplural value (no link and no check for existence).

```

4291 \newcommand*{\glsaccesssymbolplural}[1]{\glentrysymbolplural{#1}}
```

`esssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```

4292 \newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}
```

`esssymbolplural` Display the symbolplural value (no link and no check for existence). converted to upper case.

```

4293 \newcommand*{\GLSaccesssymbolplural}[1]{%
4294 \protect\mfirstucMakeUppercase{\glentrysymbolplural{#1}}}
```

`\glsaccessdesc` Display the desc value (no link and no check for existence).

```

4295 \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.

```

4296 \newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}
```

`\GLSaccessdesc` Display the desc value (no link and no check for existence). converted to upper case.  
4297 `\newcommand*{\GLSaccessdesc}[1]{%`  
4298 `\protect\mfirstucMakeUppercase{\glentrydesc{#1}}}`

`ccessdescplural` Display the descplural value (no link and no check for existence).  
4299 `\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.  
4300 `\newcommand*{\Glsaccessdescplural}[1]{\Glentrydescplural{#1}}`

`ccessdescplural` Display the descplural value (no link and no check for existence). converted to upper case.  
4301 `\newcommand*{\GLSaccessdescplural}[1]{%`  
4302 `\protect\mfirstucMakeUppercase{\glentrydescplural{#1}}}`

`\glsaccessshort` Display the short form (no link and no check for existence).  
4303 `\newcommand*{\glsaccessshort}[1]{\glentryshort{#1}}`

`\Glsaccessshort` Display the short form with first letter converted to uppercase (no link and no check for existence).  
4304 `\newcommand*{\Glsaccessshort}[1]{\Glentryshort{#1}}`

`\GLSaccessshort` Display the short value (no link and no check for existence). converted to upper case.  
4305 `\newcommand*{\GLSaccessshort}[1]{%`  
4306 `\protect\mfirstucMakeUppercase{\glentryshort{#1}}}`

`lsaccessshortpl` Display the short plural form (no link and no check for existence).  
4307 `\newcommand*{\glsaccessshortpl}[1]{\glentryshortpl{#1}}`

`lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).  
4308 `\newcommand*{\Glsaccessshortpl}[1]{\Glentryshortpl{#1}}`

`LSaccessshortpl` Display the shortplural value (no link and no check for existence). converted to upper case.  
4309 `\newcommand*{\GLSaccessshortpl}[1]{%`  
4310 `\protect\mfirstucMakeUppercase{\glentryshortpl{#1}}}`

`\glsaccesslong` Display the long form (no link and no check for existence).  
4311 `\newcommand*{\glsaccesslong}[1]{\glentrylong{#1}}`

`\Glsaccesslong` Display the long form (no link and no check for existence).  
4312 `\newcommand*{\Glsaccesslong}[1]{\Glentrylong{#1}}`

`\GLSaccesslong` Display the long value (no link and no check for existence). converted to upper case.  
4313 `\newcommand*{\GLSaccesslong}[1]{%`  
4314 `\protect\mfirstucMakeUppercase{\glentrylong{#1}}}`

`glsaccesslongpl` Display the long plural form (no link and no check for existence).  
 4315 `\newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}`

`Glsaccesslongpl` Display the long plural form (no link and no check for existence).  
 4316 `\newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}`

`GLSaccesslongpl` Display the longplural value (no link and no check for existence). converted to upper case.  
 4317 `\newcommand*{\GLSaccesslongpl}[1]{%`  
 4318 `\protect\mfirstucMakeUppercase{\glsentrylongpl{#1}}}`  
 End of else part  
 4319 `}`

## 1.5 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.  
 4320 `\glsaddstoragekey{category}{general}{\glscategory}`

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.  
 4321 `\newcommand{\glsifcategory}[4]{%`  
 4322 `\ifglsfieldeq{#1}{category}{#2}{#3}{#4}%`  
 4323 `}`

Categories can have attributes.

`categoryattribute` `\glssetcategoryattribute{<category>}{<attribute-label>}{<value>}`

Set (or override if already set) an attribute for the given category.

4324 `\newcommand*{\glssetcategoryattribute}[3]{%`  
 4325 `\csdef{@glsxtr@categoryattr@@#1@#2}{#3}%`  
 4326 `}`

`categoryattribute` `\glsgetcategoryattribute{<category>}{<attribute-label>}`

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

4327 `\newcommand*{\glsgetcategoryattribute}[2]{%`  
 4328 `\csuse{@glsxtr@categoryattr@@#1@#2}%`  
 4329 `}`

categoryattribute `\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}`

Tests if the category has the given attribute set.

```
4330 \newcommand*{\glshascategoryattribute}[4]{%
4331   \ifcsvoid{@glstr@categoryattr@#1@#2}{#4}{#3}%
4332 }
```

\glissetattribute `\glissetattribute{<entry label>}{<attribute-label>}{<value>}`

Short cut where the category label is obtained from the entry information.

```
4333 \newcommand*{\glissetattribute}[3]{%
4334   \glsetcategoryattribute{\glscategory{#1}}{#2}{#3}%
4335 }
```

\glsggetattribute `\glsggetattribute{<entry label>}{<attribute-label>}`

Short cut where the category label is obtained from the entry information.

```
4336 \newcommand*{\glsggetattribute}[2]{%
4337   \glsgcategoryattribute{\glscategory{#1}}{#2}%
4338 }
```

\glshasattribute `\glshasattribute{<entry label>}{<attribute-label>}{<true>}{<false>}`

Short cut to test if the given attribute has been set where the category label is obtained from the entry information.

```
4339 \newcommand*{\glshasattribute}[4]{%
4340   \ifglstryexists{#1}%
4341   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
4342   {#4}%
4343 }
```

categoryattribute `\glisifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

True if category has the attribute with the given value.

```
4344 \newcommand{\glisifcategoryattribute}[5]{%
```

```

4345 \ifcsundef{@glxtr@categoryattr@@#1@#2}%
4346 {#5}%
4347 {\ifcsstring{@glxtr@categoryattr@@#1@#2}{#3}{#4}{#5}}%
4348 }

```

`\glsifattribute` `\glsifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

Short cut to determine if the given entry has a category with the given attribute set.

```

4349 \newcommand{\glsifattribute}[5]{%
4350   \ifglentryexists{#1}%
4351   {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
4352   {#5}%
4353 }

```

Set attributes for the default general category:

```
4354 \glsetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
4355 \glsetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```

4356 \newcommand*{\glsetregularcategory}[1]{%
4357   \glsetcategoryattribute{#1}{regular}{true}%
4358 }

```

`ifregularcategory` `\glsifregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```

4359 \newcommand{\glsifregularcategory}[3]{%
4360   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
4361 }

```

`notregularcategory` `\glsifnotregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```

4362 \newcommand{\glsifnotregularcategory}[3]{%
4363   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
4364 }

```



`\glsifregular` `\glsifregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to true.

```
4365 \newcommand{\glsifregular}[3]{%
4366   \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
4367 }
```

`\glsifnotregular` `\glsifnotregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to false.

```
4368 \newcommand{\glsifnotregular}[3]{%
4369   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
4370 }
```

`\glsforeachincategory` `\glsforeachincategory[<glossary labels>]{<category-label>}{<glossary-cs>}{<label-cs>}{<body>}`

Iterates through all entries in all the glossaries (or just those listed in *<glossary labels>*) and does *<body>* if the category matches *<category-label>*. The control sequences *<glossary-cs>* and *<label-cs>* may be used in *<body>* to access the glossary label and entry label for the current iteration.

```
4371 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
4372   \forallglossaries[#1]{#3}%
4373   {%
4374     \forlentries[#3]{#4}%
4375     {%
4376       \glsifcategory{#4}{#2}{#5}{}%
4377     }%
4378   }%
4379 }
```

`\glsforeachwithattribute` `\glsforeachwithattribute[<glossary labels>]{<attribute-label>}{<attribute-value>}{<glossary-cs>}{<label-cs>}{<body>}`

Iterates through all entries in all the glossaries (or just those listed in *<glossary labels>*) and does *<body>* if the category attribute *<attribute-label>* matches *<attribute-value>*. The control sequences *<glossary-cs>* and *<label-cs>* may be used in *<body>* to access the glossary label and entry label for the current iteration.

```

4380 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
4381   \forallglossaries[#1]{#4}%
4382   {%
4383     \forglsentries[#4]{#5}%
4384     {%
4385       \glsifattribute{#5}{#2}{#3}{#6}{}%
4386     }%
4387   }%
4388 }

```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to `index` and add `\glstrpostdescription`.

```

4389 \ifdef\newterm
4390 {%

```

`\newterm`

```

4391   \renewcommand*{\newterm}[2][ ]{%
4392     \newglossaryentry{#2}%
4393     {type={index},category=index,name={#2},%
4394      description={\glstrpostdescription\nopostdesc},#1}%
4395   }

```

Indexed terms are regular by default.

```

4396   \glsssetcategoryattribute{index}{regular}{true}

```

`trpostdescindex`

```

4397   \newcommand*{\glstrpostdescindex}{}
4398 }
4399 {}

```

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`.

```

4400 \ifdef\printsymbols
4401 {%

```

`glstrnewsymbol` Unlike `\newterm`, this has a separate argument for the label (since the symbol will likely contain commands).

```

4402   \newcommand*{\glstrnewsymbol}[3][ ]{%
4403     \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
4404   }

```

Symbols are regular by default.

```

4405   \glsssetcategoryattribute{symbol}{regular}{true}

```

`trpostdescsymbol`

```

4406   \newcommand*{\glstrpostdescsymbol}{}

```

```
4407 }
4408 {}
```

Similar for the numbers option.

```
4409 \ifdef\printnumbers
4410 {%
```

glsxtrnewnumber

```
4411 \ifdef\printnumbers
4412   \newcommand*\glsxtrnewnumber[3][]{%
4413     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
4414   }
```

Numbers are regular by default.

```
4415   \glssetcategoryattribute{number}{regular}{true}
```

rpostdescnumber

```
4416   \newcommand*\glsxtrpostdescnumber{}

4417 }
4418 {}
```

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.

```
4419 \newcommand*\glsxtrsetcategory[2]{%
4420   \@for\@glsxtr@label:=#1\do
4421   {%
4422     \glsfieldxdef{\@glsxtr@label}{category}{#2}%
4423   }%
4424 }
```

categoryforall Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```
4425 \newcommand*\glsxtrsetcategoryforall[2]{%
4426   \forallglossaries[#1]{\@glsxtr@type}{%
4427     \forglsentries[\@glsxtr@type]{\@glsxtr@label}%
4428     {%
4429       \glsfieldxdef{\@glsxtr@label}{category}{#2}%
4430     }%
4431   }%
4432 }
```

trfieldtitlecase

```
\glsxtrfieldtitlecase{<label>}{<field>}
```

Apply title casing to the contents of the given field.

```
4433 \newcommand*\glsxtrfieldtitlecase[2]{%
```

```

4434 \expandafter\glxtrfieldtitlecasecs\expandafter
4435   {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
4436 }

```

fieldtitlecasecs The command used by \glxtrfieldtitlecase. May be redefined to use a different command, for example, \xcapitalisefmtwords.

```

4437 \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.

```

4438 \ifpackageloaded{glossaries-accsupp}
4439 {
4440   \renewcommand*{\glossentrydesc}[1]{%
4441     \glsdoifexistsorwarn{#1}%
4442     {%
4443       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```

4444   \glshasattribute{#1}{glossdescfont}%
4445   {%
4446     \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
4447     \ifcsdef{\@glxtr@attrval}%
4448     {%
4449       \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
4450     }%
4451     {%
4452       \GlossariesExtraWarning{Unknown control sequence name
4453         ‘\@glxtr@attrval’ supplied in glossdescfont attribute
4454         for entry ‘#1’. Ignoring}%
4455       \let\@glxtr@glossdescfont\@firstofone
4456     }%
4457   }%
4458   {\let\@glxtr@glossdescfont\@firstofone}%
4459   \glsifattribute{#1}{glossdesc}{firstuc}%
4460   {%
4461     \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
4462   }%
4463   {%
4464     \glsifattribute{#1}{glossdesc}{title}%
4465     {%
4466       \@glxtr@do@titlecaps@warn
4467       \glsdescriptionaccessdisplay
4468     {%
4469       \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
4470     }%
4471     {#1}%

```

```

4472     }%
4473     {%
4474         \@glxtr@glossdescfont{\glsaccessdesc{#1}}}%
4475     }%
4476 }%
4477 }%
4478 }
4479 }
4480 {
4481 \renewcommand*{\glossentrydesc}[1]{%
4482 \glsdoifexistsorwarn{#1}%
4483 {%
4484 \glssetabbrvfmt{\glscategory{#1}}%
4485 \glsattribute{#1}{glossdescfont}%
4486 {%
4487 \edef\@glxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
4488 \ifcsdef{\@glxtr@attrval}%
4489 {%
4490 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
4491 }%
4492 {%
4493 \GlossariesExtraWarning{Unknown control sequence name
4494 '\@glxtr@attrval' supplied in glossdescfont attribute
4495 for entry '#1'. Ignoring}%
4496 \let\@glxtr@glossdescfont\@firstofone
4497 }%
4498 }%
4499 {\let\@glxtr@glossdescfont\@firstofone}%
4500 \glsifattribute{#1}{glossdesc}{firstuc}%
4501 {%
4502 \@glxtr@glossdescfont{\Glsentrydesc{#1}}}%
4503 }%
4504 {%
4505 \glsifattribute{#1}{glossdesc}{title}%
4506 {%
4507 \@glxtr@do@titlecaps@warn
4508 \@glxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
4509 }%
4510 {%
4511 \@glxtr@glossdescfont{\glsentrydesc{#1}}}%
4512 }%
4513 }%
4514 }%
4515 }
4516 }

```

`\glossentryname` If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

4517 \ifpackage{glossaries-accsupp}

```

```

4518 {
4519   \renewcommand*{\glossentryname}[1]{%
4520     \@glsdoifexistsorwarn{#1}%
4521     {%
4522       \glsetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

4523     \glshasattribute{#1}{glossnamefont}%
4524     {%
4525       \edef\@glsextr@attrval{\glsetattribute{#1}{glossnamefont}}%
4526       \ifcsdef{\@glsextr@attrval}%
4527       {%
4528         \letcs{\@glsextr@glossnamefont}{\@glsextr@attrval}%
4529       }%
4530     }%
4531     \GlossariesExtraWarning{Unknown control sequence name
4532       ‘\@glsextr@attrval’ supplied in glossnamefont attribute
4533       for entry ‘#1’. Reverting to default \string\glsnamefont}%
4534     \let\@glsextr@glossnamefont\glsnamefont
4535   }%
4536 }%
4537 {\let\@glsextr@glossnamefont\glsnamefont}%
4538 \glsifattribute{#1}{glossname}{firstuc}%
4539 {%
4540   \glsnameaccessdisplay
4541   {%
4542     \@glsextr@glossnamefont{\Glsentryname{#1}}%
4543   }%
4544   {#1}%
4545 }%
4546 {%
4547   \glsifattribute{#1}{glossname}{title}%
4548   {%
4549     \@glsextr@do@titlecaps@warn
4550     \glsnameaccessdisplay
4551     {%
4552       \@glsextr@glossnamefont{\glsextr@fieldtitlecase{#1}{name}}%
4553     }%
4554     {#1}%
4555   }%
4556   {%
4557     \glsifattribute{#1}{glossname}{uc}%
4558     {%
4559       \glsnameaccessdisplay
4560       {%

```

Hide the label from the upper-casing command.

```

4561       \letcs{\glo@name}{glo\@glsetoklabel{#1}@name}%
4562       \@glsextr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
4563     }%

```

```

4564         {#1}%
4565     }%
4566     {%
4567         \letcs{\glo@name}{\glo@\glsdetoklabel{#1}@name}%
4568         \glsnameaccessdisplay
4569     {%
4570         \expandafter\@glxtr@glossnamefont\expandafter{\glo@name}%
4571     }%
4572     {#1}%
4573 }%
4574 }%
4575 }%

```

Do post-name hook:

```

4576     \glxtrpostnamehook{#1}%
4577 }%
4578 }
4579 }
4580 {
4581     \renewcommand*{\glossentryname}[1]{%
4582         \@glsdoifexistsorwarn{#1}%
4583     {%
4584         \glsetabbrvfmt{\glscategory{#1}}%
4585         \glshasattribute{#1}{\glossnamefont}%
4586     {%
4587         \edef\@glxtr@attrval{\glsetattribute{#1}{\glossnamefont}}%
4588         \ifcsdef{\@glxtr@attrval}%
4589         {%
4590             \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
4591         }%
4592         {%
4593             \GlossariesExtraWarning{Unknown control sequence name
4594             '\@glxtr@attrval' supplied in glossnamefont attribute
4595             for entry '#1'. Reverting to default \string\glsnamefont}%
4596             \let\@glxtr@glossnamefont\glsnamefont
4597         }%
4598     }%
4599     {\let\@glxtr@glossnamefont\glsnamefont}%
4600     \gl@ifattribute{#1}{\glossname}{\firstuc}%
4601     {%
4602         \@glxtr@glossnamefont{\Glsentryname{#1}}%
4603     }%
4604     {%
4605         \gl@ifattribute{#1}{\glossname}{\title}%
4606     {%
4607         \@glxtr@do@titlecaps@warn
4608         \@glxtr@glossnamefont{\glxtrfieldtitlecase{#1}{name}}%
4609     }%
4610     {%
4611         \gl@ifattribute{#1}{\glossname}{\uc}%

```

```
4612         {%
```

Hide the label from the upper-casing command.

```
4613         \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
4614         \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
4615         }%
4616         {%
```

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.

```
4617         \letcs{\glo@name}{glo@glstdetoklabel{#1}@name}%
4618         \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
4619         }%
4620         }%
4621         }%
```

Do post-name hook.

```
4622         \glstrpostnamehook{#1}%
4623         }%
4624     }
4625 }
```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```
4626 \ifpackageloaded{glossaries-accsupp}
4627 {
4628     \renewcommand*{\Glossentryname}[1]{%
4629         \@glstdoifexistsorwarn{#1}%
4630         {%
4631             \glsetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the `glossnamefont` attribute to determine the font applied.

```
4632         \glshasattribute{#1}{glossnamefont}%
4633         {%
4634             \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
4635             \ifcsdef{\@glstr@attrval}%
4636             {%
4637                 \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
4638                 }%
4639             {%
4640                 \GlossariesExtraWarning{Unknown control sequence name
4641                     ‘\@glstr@attrval’ supplied in glossnamefont attribute
4642                     for entry ‘#1’. Reverting to default \string\glsnamefont}%
4643                 \let\@glstr@glossnamefont\glsnamefont
4644                 }%
4645             }%
4646             {\let\@glstr@glossnamefont\glsnamefont}%
4647             \glsnameaccessdisplay
4648             {%
4649                 \@glstr@glossnamefont{\Glsentryname{#1}}%
4650             }%
4651             {#1}%
```



Do post-name hook:

```
4652 \glxtrpostnamehook{#1}%
4653 }%
4654 }
4655 }
4656 {
4657 \renewcommand*{\Glossentryname}[1]{%
4658 \@glxdoifexistsorwarn{#1}%
4659 {%
4660 \glsetabbrvfmt{\glscategory{#1}}%
4661 \glshasattribute{#1}{glossnamefont}%
4662 {%
4663 \edef\@glxtr@attrval{\glsetattribute{#1}{glossnamefont}}%
4664 \ifcsdef{\@glxtr@attrval}%
4665 {%
4666 \letcs{\@glxtr@glossnamefont}{\@glxtr@attrval}%
4667 }%
4668 {%
4669 \GlossariesExtraWarning{Unknown control sequence name
4670 '\@glxtr@attrval' supplied in glossnamefont attribute
4671 for entry '#1'. Reverting to default \string\glnamefont}%
4672 \let\@glxtr@glossnamefont\glnamefont
4673 }%
4674 }%
4675 {\let\@glxtr@glossnamefont\glnamefont}%
4676 \@glxtr@glossnamefont{\Glsentryname{#1}}%
```

Do post-name hook:

```
4677 \glxtrpostnamehook{#1}%
4678 }%
4679 }
4680 }
```

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.

```
4681 \newcommand*{\glxtrpostnamehook}[1]{%
4682 \def\@glxnumberformat{glxnumberformat}%
4683 \glxtrdoautoindexname{#1}{indexname}%
```

Allow categories to hook in here.

```
4684 \csuse{glxtrpostname\glscategory{#1}}%
4685 }
```

`\glxtrformat@override` Determines if the format key should override the indexing attribute value.

```
4686 \newif\if@glxtr@format@override
4687 \@glxtr@format@overridefalse
```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

4688 \ifpackageloaded{hyperref}
4689 {
    If hyperref's hyperindex option is on, then hyperref will automatically add \hyperpage, so
    don't add it.
4690   \ifHy@hyperindex
4691     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
4692       \@glstr@format@overridetrue
4693       \appto\theindex{\let\glshypernumber\@firstofone}%
4694     }
4695   \else
4696     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
4697       \@glstr@format@overridetrue
4698       \appto\theindex{\let\glshypernumber\hyperpage}%
4699     }
4700   \fi
4701 }
4702 {
4703   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
4704     \@glstr@format@overridetrue
4705   }
4706 }
4707 \@onlypreamble\GlsXtrEnableIndexFormatOverride

```

`doautoindexname`

```

4708 \newcommand*{\glstrdoautoindexname}[2]{%
4709   \glshasattribute{#1}{#2}%
4710   {%
    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.
4711     \@glstr@autoindex@setname{#1}%
    If the attribute value is simply "true" don't add an encap, otherwise use the value as the encap.
4712     \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
4713     \if@glstr@format@override
4714       \ifdefstring{\@glstrnumberformat}{\glstrnumberformat}{}%
4715       {\let\@glstr@attrval\@glstrnumberformat}%
4716     \fi
4717     \ifdefstring{\@glstr@attrval}{true}%
4718     {}%
4719     {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
4720     \expandafter\glstrautoindex\expandafter{\@glo@name}%
4721   }%
4722   {}%
4723 }

```

glxstrautoindex

```
4724 \newcommand*{\glxstrautoindex}{\index}
```

toindex@setname Assign \@glo@name for use with indexname attribute.

```
4725 \newcommand*{\@glxstr@autoindex@setname}[1]{%
4726   \protected@edef\@glo@name{\glxstrautoindexentry{#1}}%
4727   \glxstrautoindexassignsort{\@glo@sort}{#1}%
4728   \@gls@checkmkidxchars\@glo@sort
4729   \@glxstr@autoindex@doextra@esc\@glo@sort
4730   \epreto\@glo@name{\@glo@sort\@glxstr@autoindex@at}%
4731 }
```

rautoindexentry Command used for the actual part when auto-indexing.

```
4732 \newcommand*{\glxstrautoindexentry}[1]{\string\glxentryname{#1}}
```

trautoindexsort Used to assign the sort value when auto-indexing.

```
4733 \newcommand*{\glxstrautoindexassignsort}[2]{%
4734   \glsletentryfield{#1}{#2}{sort}%
4735 }
```

dex@doextra@esc

```
4736 \newcommand*{\@glxstr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
4737   \ifx\@glxstr@autoindex@esc\@gls@quotechar
4738   \else
4739     \def\@gls@checkedmkidx{%
4740       \edef\@glxstr@checkspch{%
4741         \noexpand\@glxstr@autoindex@escquote\expandonce{#1}%
4742         \noexpand\@empty\@glxstr@autoindex@esc\noexpand\@nnil
4743         \@glxstr@autoindex@esc\noexpand\@empty\noexpand\@glxstr@endescspch}%
4744       \@glxstr@checkspch
4745       \let#1\@gls@checkedmkidx\relax
4746     \fi
```

Escape actual character unless it has already been escaped.

```
4747   \ifx\@glxstr@autoindex@at\@gls@actualchar
4748   \else
4749     \def\@gls@checkedmkidx{%
4750       \edef\@glxstr@checkspch{%
4751         \noexpand\@glxstr@autoindex@escat\expandonce{#1}%
4752         \noexpand\@empty\@glxstr@autoindex@at\noexpand\@nnil
4753         \@glxstr@autoindex@at\noexpand\@empty\noexpand\@glxstr@endescspch}%
4754       \@glxstr@checkspch
4755       \let#1\@gls@checkedmkidx\relax
4756     \fi
```

Escape level character unless it has already been escaped.

```
4757   \ifx\@glxstr@autoindex@level\@gls@levelchar
4758   \else
```

```

4759 \def\@gls@checkedmkidx{%
4760 \edef\@@glsxtr@checkspch{%
4761 \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
4762 \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
4763 \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
4764 \@@glsxtr@checkspch
4765 \let#1\@gls@checkedmkidx\relax
4766 \fi

```

Escape encap character unless it has already been escaped.

```

4767 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
4768 \else
4769 \def\@gls@checkedmkidx{%
4770 \edef\@@glsxtr@checkspch{%
4771 \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
4772 \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
4773 \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
4774 \@@glsxtr@checkspch
4775 \let#1\@gls@checkedmkidx\relax
4776 \fi
4777 }

```

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.

```

4778 \newcommand*{\@glsxtr@autoindex@at}{}

```

trSetActualChar Set the actual character.

```

4779 \newcommand*{\GlsXtrSetActualChar}[1]{%
4780 \gdef\@glsxtr@autoindex@at{#1}%
4781 \def\@glsxtr@autoindex@escat##1##2##3\@glsxtr@endescspch{%
4782 \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
4783 }%
4784 }
4785 \@onlypreamble\GlsXtrSetActualChar
4786 \makeatother
4787 \GlsXtrSetActualChar{@}
4788 \makeatletter

```

autoindex@encap Encap character for use with \index.

```

4789 \newcommand*{\@glsxtr@autoindex@encap}{}

```

XtrSetEncapChar Set the encap character.

```

4790 \newcommand*{\GlsXtrSetEncapChar}[1]{%
4791 \gdef\@glsxtr@autoindex@encap{#1}%
4792 \def\@glsxtr@autoindex@escencap##1##2##3\@glsxtr@endescspch{%
4793 \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
4794 }%

```

```

4795 }
4796 \GlsXtrSetEncapChar{||}
4797 \@onlypreamble\GlsXtrSetEncapChar

autoindex@level  Level character for use with \index.
4798 \newcommand*{\@glxtr@autoindex@level}{%}

XtrSetLevelChar  Set the encap character.
4799 \newcommand*{\GlsXtrSetLevelChar}[1]{%
4800   \gdef\@glxtr@autoindex@level{#1}%
4801   \def\@glxtr@autoindex@esclevel##1#1##2#1##3\@glxtr@endescspch{%
4802     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@esclevel}{##1}{##2}{##3}%
4803   }%
4804 }
4805 \GlsXtrSetLevelChar{!}
4806 \@onlypreamble\GlsXtrSetLevelChar

r@autoindex@esc  Escape character for use with \index.
4807 \newcommand*{\@glxtr@autoindex@esc}{"}

lsXtrSetEscChar  Set the escape character.
4808 \newcommand*{\GlsXtrSetEscChar}[1]{%
4809   \gdef\@glxtr@autoindex@esc{#1}%
4810   \def\@glxtr@autoindex@escquote##1#1##2#1##3\@glxtr@endescspch{%
4811     \@glxtr@autoindex@escspch{#1}{\@glxtr@autoindex@escquote}{##1}{##2}{##3}%
4812   }%
4813 }
4814 \GlsXtrSetEscChar{"}
4815 \@onlypreamble\GlsXtrSetEscChar

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:
4816 \ifdef\actualchar
4817 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
4818 {}

Quote character \quotechar:
4819 \ifdef\quotechar
4820 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
4821 {}

Level character \levelchar:
4822 \ifdef\levelchar
4823 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
4824 {}

Encap character \encapchar:
4825 \ifdef\encapchar
4826 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
4827 {}

```

leto@endescspch

```
4828 \def\@glxtr@gobbleto@endescspch#1\@glxtr@endescspch{}
```

toindex@esc@spch

```
\@glxtr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}
```

```
4829 \newcommand*{\@glxtr@autoindex@escspch}[5]{%
4830   \@glstmpb=\expandafter{\@glstmpb\the\toks@}%
4831   \toks@={#3}%
4832   \ifx\@nnil#3\relax
4833     \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch#5\@glxtr@endescspch}%
4834   \else
4835     \ifx\@nnil#4\relax
4836       \edef\@glstmpb{\the\@glstmpb\the\toks@}%
4837       \def\@glxtr@checkspch{\@glxtr@gobbleto@endescspch
4838         #4#5\@glxtr@endescspch}%
4839     \else
4840       \edef\@glstmpb{\the\@glstmpb\the\toks@
4841         \@glxtr@autoindex@esc#1}%
4842       \def\@glxtr@checkspch{#2#5#1\@nnil#1\@glxtr@endescspch}%
4843     \fi
4844   \fi
4845   \@glxtr@checkspch
4846 }
```

\Glossentrydesc   Redefine to set the abbreviation format and accessibility support.

```
4847 \renewcommand*{\Glossentrydesc}[1]{%
4848   \glsoifexistsorwarn{#1}%
4849   {%
4850     \glsssetabbrvfmt{\glscategory{#1}}%
4851     \glssaccessdesc{#1}%
4852   }%
4853 }
```

\lossentrysymbol   Redefine to set the abbreviation format and accessibility support.

```
4854 \renewcommand*{\lossentrysymbol}[1]{%
4855   \glsoifexistsorwarn{#1}%
4856   {%
4857     \glsssetabbrvfmt{\glscategory{#1}}%
4858     \glssaccesssymbol{#1}%
4859   }%
4860 }
```

\lossentrysymbol   Redefine to set the abbreviation format and accessibility support.

```
4861 \renewcommand*{\lossentrysymbol}[1]{%
4862   \glsoifexistsorwarn{#1}%
4863   {%
```

```

4864 \glssetabbrvfmt{\glscategory{#1}}%
4865 \Glsaccesssymbol{#1}%
4866 }%
4867 }

```

Allow initials to be marked but only use the formatting for the tag in the glossary.

`\enableInitialTagging` Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```

4868 \newcommand*{\GlsXtrEnableInitialTagging}{%
4869 \ifstar\s@glxtr@enabletagging\glxtr@enabletagging
4870 }
4871 \@onlypreamble\GlsXtrEnableInitialTagging

```

`\r@enabletagging` Starred version undefines command.

```

4872 \newcommand*{\s@glxtr@enabletagging}[2]{%
4873 \undef#2%
4874 \@glxtr@enabletagging{#1}{#2}%
4875 }

```

`\r@enabletagging` Internal command.

```

4876 \newcommand*{\@glxtr@enabletagging}[2]{%
    Set attributes for categories given in the first argument.
4877 \@for\@glxtr@cat:=#1\do
4878 {%
4879 \ifdefempty\@glxtr@cat
4880 {}%
4881 {\glssetcategoryattribute{\@glxtr@cat}{tagging}{true}}%
4882 }%
4883 \newrobustcmd*#2[1]{##1}%
4884 \def\@glxtr@taggingcs{#2}%
4885 \renewcommand*\@glxtr@activate@initialtagging{%
4886 \let#2\@glxtr@tag
4887 }%
4888 \ifundef\@gls@preglossaryhook
4889 {\GlossariesExtraWarning{Initial tagging requires at least
4890 glossaries.sty v4.19 to work correctly}}%
4891 {}%
4892 }

```

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`

```

4893 \ifundef\mfu@checkword@do
4894 {
4895 \newcommand*{\mfu@checkword@do}[1]{%

```

```

4896 \ifdefstring{\mfu@checkword@arg}{#1}%
4897 {%
4898     \let\@mfu@domakefirstuc\@firstofone
4899     \listbreak
4900 }%
4901 {}%
4902 }

```

`\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.

```

4903 \ifundef\mfu@checkword
4904 {
4905     \newcommand{\@glxtr@do@titlecaps@warn}{%
4906         \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
4907             support not available}%
4908         \let\@glxtr@do@titlecaps@warn\relax
4909     }
4910 }
4911 {
4912     \renewcommand*{\mfu@checkword}[1]{%
4913         \def\mfu@checkword@arg{#1}%
4914         \let\@mfu@domakefirstuc\makefirstuc
4915         \forlistloop\mfu@checkword@do\@mfu@nocaplist
4916     }
4917 }
4918 }
4919 {}% no patch required

```

`@titlecaps@warn` Do warning if title case not supported.

```

4920 \newcommand*{\@glxtr@do@titlecaps@warn}{}

```

`@initialtagging` Used in `\printglossary` but at least `v4.19` of `glossaries` required.

```

4921 \newcommand*{\@glxtr@activate@initialtagging}{}

```

`\@glxtr@tag` Definition of tagging command when used in glossary.

```

4922 \newrobustcmd*{\@glxtr@tag}[1]{%
4923     \glxtr@tagfont{#1}{#1}%
4924     \glxtr@tagfont{#1}{#1}%
4925 }

```

`\glxtrtagfont` Used in the glossary.

```

4926 \newcommand*{\glxtrtagfont}[1]{\underline{#1}}

```

`preglossaryhook` This macro was introduced in `glossaries` version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.



```
4927 \ifdef \@gls@preglossaryhook
```

```
4928 {
```

```
4929   \renewcommand*{\@gls@preglossaryhook}{%
```

```
4930     \@glsxtr@activate@initialtagging
```

Since the glossaries are automatically scoped, \@glsxtr@org@postdescription shouldn't already be defined, but check anyway just as a precautionary measure.

```
4931   \ifundef \@glsxtr@org@postdescription
```

```
4932   {%
```

```
4933     \let \@glsxtr@org@postdescription \glspostdescription
```

```
4934     \renewcommand*{\glspostdescription}{%
```

```
4935       \ifglsentryexists{\glscurrententrylabel}%
```

```
4936       {%
```

```
4937         \glsxtrpostdescription
```

```
4938         \@glsxtr@org@postdescription
```

```
4939       }%
```

```
4940     }%
```

```
4941   }%
```

```
4942 }%
```

```
4943 {}%
```

Enable the options used by \@glsxtrp:

```
4944   \glossxtrsetpopts
```

```
4945 }%
```

```
4946 }
```

```
4947 {}
```

`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.

```
4948 \newcommand*{\glsxtrpostdescription}{%
```

```
4949   \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
```

```
4950 }
```

`postdescgeneral`

```
4951 \newcommand*{\glsxtrpostdescgeneral}{{}
```

`xtrpostdescterm`

```
4952 \newcommand*{\glsxtrpostdescterm}{{}
```

`postdescacronym`

```
4953 \newcommand*{\glsxtrpostdescacronym}{{}
```

`descabbreviation`

```
4954 \newcommand*{\glsxtrpostdescabbreviation}{{}
```

`glspostlinkhook` Redefine the post link hook used by commands like \gls to make it easier for categories or attributes to modify this action. Since this hook occurs outside the existence check of

commands like `\gls`, this needs to be checked again here. Do nothing if the entry hasn't been defined.

```
4955 \renewcommand*{\glspostlinkhook}{%
4956   \ifglentryexists{\glslabel}{\glstrpostlinkhook}{}%
4957 }
```

`\glstrpostlinkhook` The entry label should already be stored in `\glslabel` by `\@gls@link`.

```
4958 \newcommand*{\glstrpostlinkhook}{%
4959   \glstrdiscardperiod{\glslabel}%
4960   {\glstrpostlinkendsentence}%
4961   {\glstrpostlink}%
4962 }
```

`\glstrpostlink`

```
4963 \newcommand*{\glstrpostlink}{%
4964   \csuse{glstrpostlink\glscategory{\glslabel}}%
4965 }
```

`\linkendsentence` Done by `\glstrpostlinkhook` if a full stop is discarded.

```
4966 \newcommand*{\glstrpostlinkendsentence}{%
4967   \ifcsdef{glstrpostlink\glscategory{\glslabel}}
4968   {%
4969     \csuse{glstrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
4970   .\spacefactor\sfcode'\. \relax
4971   }%
4972   {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
4973   \spacefactor\sfcode'\. \relax
4974   }%
4975 }
```

`\dDescOnFirstUse` Provide a command for appending the description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
4976 \newcommand*{\glstrpostlinkAddDescOnFirstUse}{%
4977   \glstrifwasfirstuse{\space(\glssaccessdesc{\glslabel})}{}%
4978 }
```

`\ymbolOnFirstUse` Provide a command for appending the symbol (if defined) in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
4979 \newcommand*{\glstrpostlinkAddSymbolOnFirstUse}{%
4980   \glstrifwasfirstuse
4981   {%
4982     \ifglshassymbol{\glslabel}{\space(\glssaccesssymbol{\glslabel})}{}%
4983   }%
4984   {}%
4985 }
```

`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).

```

4986 \newcommand*{\glxtrdiscardperiod}[3]{%
4987   \glxtrifwasfirstuse
4988   {%
4989     \glusifattribute{#1}{retainfirstuseperiod}{true}%
4990     {#3}%
4991     {%
4992       \glusifattribute{#1}{discardperiod}{true}%
4993       {%
4994         \glusifplural
4995         {%
4996           \glusifattribute{#1}{pluraldiscardperiod}{true}%
4997           {\glxtrifperiod{#2}{#3}}%
4998           {#3}%
4999         }%
5000       }%
5001       \glxtrifperiod{#2}{#3}%
5002     }%
5003   }%
5004   {#3}%
5005 }%
5006 }%
5007 {%
5008   \glusifattribute{#1}{discardperiod}{true}%
5009   {%
5010     \glusifplural
5011     {%
5012       \glusifattribute{#1}{pluraldiscardperiod}{true}%
5013       {\glxtrifperiod{#2}{#3}}%
5014       {#3}%
5015     }%
5016     {%
5017       \glxtrifperiod{#2}{#3}%
5018     }%
5019   }%
5020   {#3}%
5021 }%
5022 }

```

`\glxtrifperiod` Make a convenient user command to check if the next character is a full stop (period). Works like `\@ifstar` but uses `\new@ifnextchar` rather than `\@ifnextchar`

```

5023 \newcommand*{\glxtrifperiod}[1]{\new@ifnextchar.{\@firstoftwo{#1}}}

```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

`glxtr@punclist` List of characters identified as punctuation marks. (Be careful of babel shorthands!) This

doesn't allow for punctuation marks made up from multiple characters (such as ' ').

```
5024 \newcommand*{\glxtr@punclist}{.,:;!}
```

punctuationmark Add character to punctuation list.

```
5025 \newcommand*{\glxtraddpunctuationmark}[1]{\appto\glxtr@punclist{#1}}
```

punctuationmarks Reset the punctuation list.

```
5026 \newcommand*{\glxtrsetpunctuationmarks}[1]{\def\glxtr@punclist{#1}}
```

```
\glxtrifpunc \glxtrifnextpunc{<true part>}{<false part>}
```

Test if this is followed by a punctuation mark. (Adapted from \new@ifnextchar.)

```
5027 \newcommand*{\glxtrifnextpunc}[2]{%
5028   \def\reserved@a{#1}%
5029   \def\reserved@b{#2}%
5030   \futurelet\@glspunc@token\glxtr@ifnextpunc
5031 }
```

glxtr@ifnextpunc

```
5032 \newcommand*{\glxtr@ifnextpunc}{%
5033   \glxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
5034   \reserved@b
5035 }
```

glxtr@ifpunctoken Test if the token given in the first argument is in the punctuation list.

```
5036 \newcommand*{\glxtr@ifpunctoken}[1]{%
5037   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
5038 }
```

glxtr@ifpunctoken

```
5039 \def\@glxtr@ifpunctoken#1#2{%
5040   \let\reserved@d=#2%
5041   \ifx\reserved@d\@nnil
5042     \let\glxtr@next\@glxtr@notfoundinlist
5043   \else
5044     \ifx#1\reserved@d
5045       \let\glxtr@next\@glxtr@foundinlist
5046     \else
5047       \let\glxtr@next\@glxtr@ifpunctoken
5048     \fi
5049   \fi
5050   \glxtr@next#1%
5051 }
```

glxtr@foundinlist

```
5052 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
5053 \def\@glxstr@notfoundinlist#1{\@secondoftwo}
```

glxstrdopostpunc

```
\glxstrdopostpunc{<code>}
```

If this is followed by a punctuation character, do <code> after the character otherwise do <code> before whatever comes next.

```
5054 \newcommand{\glxstrdopostpunc}[1]{%
5055   \glxstrifnextpunc{\@glxstr@swaptwo{#1}}{#1}%
5056 }
```

@glxstr@swaptwo

```
5057 \newcommand{\@glxstr@swaptwo}[2]{#2#1}
```

## 1.6 Abbreviations

The “acronym” code from glossaries is misnamed as it’s more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there’s a style for the given category, apply it.

```
5058 \define@key{glxstrabbrv}{category}{%
5059   \edef\glscategorylabel{#1}%
5060   \ifcsdef{@glxstr@current@#1}%
5061   {%
```

Warning should already have been issued.

```
5062   \let\@glxstr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
5063   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
5064   \glxstr@applyabbrvstyle{\csname @glxstr@current@#1\endcsname}%
5065   \let\GlsXtrWarnDeprecatedAbbrStyle\@glxstr@orgwarndep
5066 }%
5067 {}%
5068 }
```

Save the short plural form. This may be needed before the entry is defined.

```
5069 \define@key{glxstrabbrv}{shortplural}{%
5070   \def\@glx@shortpl{#1}%
5071 }
```

Similarly for the long plural form.

```
5072 \define@key{glxstrabbrv}{longplural}{%
5073   \def\@glx@longpl{#1}%
5074 }
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

`\glsshortpltok`

```
5075 \newtoks\glsshortpltok
```

`\glslongpltok`

```
5076 \newtoks\glslongpltok
```

`sxtr@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.

```
5077 \newcommand*{\@glsxtr@insertdots}[2]{%
5078   \def#1{%
5079     \@glsxtr@insert@dots#1#2\@nnil
5080   }}
```

`xtr@insert@dots`

```
5081 \newcommand*{\@glsxtr@insert@dots}[2]{%
5082   \ifx\@nnil#2\relax
5083   \let\@glsxtr@insert@dots@next\@gobble
5084   \else
5085     \ifx\relax#2\relax
5086     \else
5087       \appto#1{#2.}%
5088     \fi
5089     \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
5090     \fi
5091     \@glsxtr@insert@dots@next#1%
5092   }}
```

`newabbreviation` Define a new generic abbreviation.

```
5093 \newcommand*{\newabbreviation}[4] [] {%
5094   \glsxtr@newabbreviation{#1}{#2}{#3}{#4}%
5095 }
```

`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.)

```
5096 \newcommand*{\glsxtr@newabbreviation}[4] {%
5097   \glskeylisttok{#1}%
5098   \glslabeltok{#2}%
5099   \glsshorttok{#3}%
5100   \glslongtok{#4}%

```

Get the category.

```
5101 \def\glscategorylabel{abbreviation}%  
5102 \glstr@applyabbrvstyle{\@glstr@current@abbreviation}%  
5103 \setkeys*{glstrabbrv}[shortplural,longplural]{#1}%
```

Set the default long plural

```
5104 \def\@glstr@longpl{#4\glstrpluralsuffix}%
```

Has the insertdots attribute been set?

```
5105 \glstrifcategoryattribute{\glscategorylabel}{insertdots}{true}%  
5106 {%  
5107 \glstr@insertdots\@glstr@short{#3}%  
5108 \expandafter\glstrshorttok\expandafter{\@glstr@short\spacefactor1000 \relax}%  
5109 \glstrifcategoryattribute{\glscategorylabel}{aposplural}{true}%  
5110 {%  
5111 \expandafter\def\expandafter\@glstr@shortpl\expandafter{\@glstr@short  
5112 '\abbrvpluralsuffix}%  
5113 }%  
5114 {%  
5115 \glstrifcategoryattribute{\glscategorylabel}{noshortplural}{true}%  
5116 {%  
5117 \let\@glstr@shortpl\@glstr@short  
5118 }%  
5119 {%  
5120 \expandafter\def\expandafter\@glstr@shortpl\expandafter{\@glstr@short  
5121 \abbrvpluralsuffix}%  
5122 }%  
5123 }%  
5124 }%  
5125 {%
```

insertdots not true.

```
5126 \glstrifcategoryattribute{\glscategorylabel}{aposplural}{true}%  
5127 {%  
5128 \def\@glstr@shortpl{#3'\abbrvpluralsuffix}%  
5129 }%  
5130 {%  
5131 \glstrifcategoryattribute{\glscategorylabel}{noshortplural}{true}%  
5132 {%  
5133 \def\@glstr@shortpl{#3}%  
5134 }%  
5135 {%  
5136 \def\@glstr@shortpl{#3\abbrvpluralsuffix}%  
5137 }%  
5138 }%  
5139 }%
```

Hook for further customisation if required:

```
5140 \glstrnewabbrvprehook{#1}{#2}{#3}%
```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary.

```
5141 \setkeys*{glstrabbrv}[category]{#1}%
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
5142 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
```

```
5143 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Do any extra setup provided by hook:

```
5144 \newabbreviationhook
```

Define this entry:

```
5145 \protected@edef\@do@newglossaryentry{%
```

```
5146 \noexpand\newglossaryentry{\the\glslabeltok}%
```

```
5147 {%
```

```
5148 type=\glstrabbrvtype,%
```

```
5149 category=abbreviation,%
```

```
5150 short={\the\glsshorttok},%
```

```
5151 shortplural={\the\glsshortpltok},%
```

```
5152 long={\the\glslongtok},%
```

```
5153 longplural={\the\glslongpltok},%
```

```
5154 name={\the\glsshorttok},%
```

```
5155 \CustomAbbreviationFields,%
```

```
5156 \the\glskeylisttok
```

```
5157 }%
```

```
5158 }%
```

```
5159 \@do@newglossaryentry
```

```
5160 \GlsXtrPostNewAbbreviation
```

```
5161 }
```

`evpresetkeyhook` Hook for extra stuff in `\newabbreviation`

```
5162 \newcommand*{\glstrnewabbrevpresetkeyhook}[3]{}%
```

`NewAbbreviation` Hook used by abbreviation styles.

```
5163 \newcommand*{\GlsXtrPostNewAbbreviation}{}%
```

`bbreviationhook` Hook for use with `\newabbreviation`.

```
5164 \newcommand*{\newabbreviationhook}{}%
```

`reviationFields`

```
5165 \newcommand*{\CustomAbbreviationFields}{}%
```

`lsxtrfullformat` Full format without case change.

```
5166 \newcommand*{\glsxtrfullformat}[2]{}%
```

```
5167 \glsfirstlongfont{\glsaccesslong{#1}}#2\glxtrfullsep{#1}%
```

```
5168 (\protect\glsfirstabbrvfont{\glsaccessshort{#1}})%
```

```
5169 }
```



`\lsxtrfullformat` Full format with case change.

```
5170 \newcommand*{\Glsxtrfullformat}[2]{%
5171   \glsfirstlongfont{\Glsaccesslong{#1}}#2\gsxtrfullsep{#1}%
5172   (\protect\glsfirstabbrvfont{\Glsaccessshort{#1}})%
5173 }
```

`\xtrfullplformat` Plural full format without case change.

```
5174 \newcommand*{\Glsxtrfullplformat}[2]{%
5175   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\gsxtrfullsep{#1}%
5176   (\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}})%
5177 }
```

`\xtrfullplformat` Plural full format with case change.

```
5178 \newcommand*{\Glsxtrfullplformat}[2]{%
5179   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\gsxtrfullsep{#1}%
5180   (\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}})%
5181 }
```

`\glxtrfullsep` Separator used by full format is a space by default. The argument is the entry's label.

```
5182 \newcommand*{\glxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with `\glentryfull` (for example, first use suppresses the long form or uses a footnote).

`\nlinefullformat` Full format without case change.

```
5183 \newcommand*{\Glsxtrinlinefullformat}{\glxtrfullformat}
```

`\nlinefullformat` Full format with case change.

```
5184 \newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}
```

`\xtrfullplformat` Plural full format without case change.

```
5185 \newcommand*{\Glsxtrinlinefullplformat}{\glxtrfullplformat}
```

`\inefullplformat` Plural full format with case change.

```
5186 \newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}
```

Redefine `\glentryfull` etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the `\glxtrfull` set of commands instead.

`\glentryfull`

```
5187 \renewcommand*{\glentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\Glsentryfull`

```
5188 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

`\glentryfullpl`

```
5189 \renewcommand*{\glentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}
```

`\Glsentryfullpl` 5190 `\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}`

`\sfirstabbrvfont` Font changing command used for the abbreviation on first use or in the full format.  
5191 `\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}`

`\bbrvdefaultfont` Font changing command used for the abbreviation on first use or in the full format.  
5192 `\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}`

`\glsabbrvfont` Font changing command used for the abbreviation on subsequent use.  
5193 `\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}`

`\bbrvdefaultfont`  
5194 `\newcommand*{\glsabbrvdefaultfont}[1]{#1}`

`\glslongfont` Font changing command used for the long form in commands like `\glsxtrlong`.  
5195 `\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}`

`\longdefaultfont` Default font changing command used for the long form in commands like `\glsxtrlong`.  
5196 `\newcommand*{\glslongdefaultfont}[1]{#1}`

`\lsfirstlongfont` Font changing command used for the long form on first use or in the full format.  
5197 `\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}`

`\longdefaultfont`  
5198 `\newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}`

`\brvpluralsuffix` Default plural suffix. Allow an alternative default suffix for abbreviations.  
5199 `\newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}`

`\brvpluralsuffix` Default plural suffix.  
5200 `\newcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}`

`\glsxtrfull` Full form (no case-change).  
5201 `\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}`  
5202 `\newcommand*\ns@glsxtrfull[2][{}]{%`  
5203 `\new@ifnextchar[{\@glsxtr@full{#1}{#2}}{%`  
5204 `{\@glsxtr@full{#1}{#2}[]}%`  
5205 `}`

`\@glsxtr@full` Low-level macro:  
5206 `\def\@glsxtr@full#1#2[#3]{%`  
5207 `\glsdoifexists{#2}%`  
5208 `{%`  
5209 `\glssetabbrvfmt{\glscategory{#2}}%`  
5210 `\let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper`  
5211 `\let\glsifplural\@secondoftwo`  
5212 `\let\gls caps case\@firstofthree`  
5213 `\let\glsinsert\@empty`  
5214 `\def\gls custom text{\glsxtrinlinefullformat{#2}{#3}}%`

What should `\glxtrifwasfirstuse` be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

```
5215 \glxtrsetupfulldefs
5216 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5217 }%
5218 \glspostlinkhook
5219 }
```

`trsetupfulldefs`

```
5220 \newcommand*{\glxtrsetupfulldefs}{%
5221 \let\glxtrifwasfirstuse\@firstoftwo
5222 }
```

`\Glsxtrfull` Full form (first letter uppercase).

```
5223 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\ns@Glsxtrfull}
5224 \newcommand*\ns@Glsxtrfull[2][{}]{%
5225 \new@ifnextchar[\@Glsxtr@full{#1}{#2}}%
5226 {\@Glsxtr@full{#1}{#2}[]}%
5227 }
```

`\@Glsxtr@full` Low-level macro:

```
5228 \def\@Glsxtr@full#1#2[#3]{%
5229 \glsdoifexists{#2}%
5230 {%
5231 \glssetabbrvfmt{\glscategory{#2}}%
5232 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5233 \let\glsifplural\@secondoftwo
5234 \let\glsupcase\@secondofthree
5235 \let\glsinsert\@empty
5236 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
5237 \glxtrsetupfulldefs
5238 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5239 }%
5240 \glspostlinkhook
5241 }
```

`\GLSxtrfull` Full form (all uppercase).

```
5242 \newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
5243 \newcommand*\ns@GLSxtrfull[2][{}]{%
5244 \new@ifnextchar[\@GLSxtr@full{#1}{#2}}%
5245 {\@GLSxtr@full{#1}{#2}[]}%
5246 }
```

`\@GLSxtr@full` Low-level macro:

```
5247 \def\@GLSxtr@full#1#2[#3]{%
5248 \glsdoifexists{#2}%
5249 }
```

```

5249 {%
5250   \glssetabbrvfmt{\glscategory{#2}}%
5251   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5252   \let\glsifplural\@secondoftwo
5253   \let\gls caps case\@thirdofthree
5254   \let\glsinsert\@empty
5255   \def\gls custom text{\mfirstucMakeUppercase{\glsxtr inline full format{#2}{#3}}}%
5256   \glsxtr setup full defs
5257   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5258 }%
5259 \gls post link hook
5260 }

```

`\glsxtrfullpl` Plural full form (no case-change).

```

5261 \newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\@ns@glsxtrfullpl}
5262 \newcommand*\ns@glsxtrfullpl[2][{}]{%
5263   \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}%
5264   {\@glsxtr@fullpl{#1}{#2}[]}%
5265 }

```

`\@glsxtr@fullpl` Low-level macro:

```

5266 \def\@glsxtr@fullpl#1#2[#3]{%
5267   \glsdoifexists{#2}%
5268   {%
5269     \glssetabbrvfmt{\glscategory{#2}}%
5270     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5271     \let\glsifplural\@firstoftwo
5272     \let\gls caps case\@firstofthree
5273     \let\glsinsert\@empty
5274     \def\gls custom text{\glsxtr inline full pl format{#2}{#3}}%
5275     \glsxtr setup full defs
5276     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5277   }%
5278   \gls post link hook
5279 }

```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```

5280 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
5281 \newcommand*\ns@Glsxtrfullpl[2][{}]{%
5282   \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
5283   {\@Glsxtr@fullpl{#1}{#2}[]}%
5284 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

5285 \def\@Glsxtr@fullpl#1#2[#3]{%
5286   \glsdoifexists{#2}%
5287   {%
5288     \glssetabbrvfmt{\glscategory{#2}}%
5289     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

```

```

5290 \let\glsifplural\@firstoftwo
5291 \let\glsifscapscase\@secondofthree
5292 \let\glsinsert\@empty
5293 \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
5294 \glsxtrsetupfulldefs
5295 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5296 }%
5297 \glspostlinkhook
5298 }

```

`\Glsxtrfullpl` Plural full form (all upper case).

```

5299 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
5300 \newcommand*\ns@Glsxtrfullpl[2][]{%
5301 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
5302 {\@Glsxtr@fullpl{#1}{#2}[]}%
5303 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

5304 \def\@Glsxtr@fullpl#1#2[#3]{%
5305 \glsdoifexists{#2}%
5306 {%
5307 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5308 \let\glsifplural\@firstoftwo
5309 \let\glsifscapscase\@thirdofthree
5310 \let\glsinsert\@empty
5311 \def\glscustomtext{%
5312 \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
5313 \glsxtrsetupfulldefs
5314 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5315 }%
5316 \glspostlinkhook
5317 }

```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```

5318 \newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\@ns@glsxtrshort}

```

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

```

5319 \newcommand*\ns@glsxtrshort[2][]{%
5320 \new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2}[]}%
5321 }

```

Read in the final optional argument:

```

5322 \def\@glsxtrshort#1#2[#3]{%
5323 \glsdoifexists{#2}%
5324 {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

5325 \glssetabbrvfmt{\glscategory{#2}}%

```

```

5326 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5327 \let\glxtrifwasfirstuse\@secondoftwo
5328 \let\gl@sifplural\@secondoftwo
5329 \let\glscapscase\@firstofthree
5330 \let\gl$insert\@empty
5331 \def\glscustomtext{%
5332     \gl@sabbrvfont{\gl@saccessshort{#2}\ifglxtrininsertinside#3\fi}%
5333     \ifglxtrininsertinside\else#3\fi
5334 }%
5335 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5336 }%
5337 \glspostlinkhook
5338 }

```

\Glsxtrshort

```

5339 \newrobustcmd*{\Glsxtrshort}{\@gl@hyp@opt\@ns@Glsxtrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
5340 \newcommand*{\ns@Glsxtrshort}[2][\@Glsxtrshort]{#1}{#2}[]{}%
5341 \new@ifnextchar[\@Glsxtrshort]{\@Glsxtrshort}{\@Glsxtrshort}{#1}{#2}[]{}%
5342 }
    Read in the final optional argument:
5343 \def\@Glsxtrshort#1#2[#3]{%
5344     \gl@doifexists{#2}%
5345     {%
5346         \gl@setabbrvfmt{\gl@category{#2}}%
5347         \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5348         \let\glxtrifwasfirstuse\@secondoftwo
5349         \let\gl@sifplural\@secondoftwo
5350         \let\glscapscase\@secondofthree
5351         \let\gl$insert\@empty
5352         \def\glscustomtext{%
5353             \gl@sabbrvfont{\gl@saccessshort{#2}\ifglxtrininsertinside#3\fi}%
5354             \ifglxtrininsertinside\else#3\fi
5355         }%
5356         \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5357     }%
5358     \glspostlinkhook
5359 }

```

\GLSxtrshort

```

5360 \newrobustcmd*{\GLSxtrshort}{\@gl@hyp@opt\@ns@GLSxtrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
5361 \newcommand*{\ns@GLSxtrshort}[2][\@GLSxtrshort]{#1}{#2}[]{}%
5362 \new@ifnextchar[\@GLSxtrshort]{\@GLSxtrshort}{\@GLSxtrshort}{#1}{#2}[]{}%
5363 }
    Read in the final optional argument:
5364 \def\@GLSxtrshort#1#2[#3]{%

```

```

5365 \glsdoifexists{#2}%
5366 {%
5367   \glsetabbrvfmt{\glscategory{#2}}%
5368   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5369   \let\glxtrifwasfirstuse\@secondoftwo
5370   \let\glsifplural\@secondoftwo
5371   \let\glscapscase\@thirdofthree
5372   \let\glsinsert\@empty
5373   \def\glscustomtext{%
5374     \mfirstucMakeUppercase
5375     {\glsabbrvfont{\glsaccessshort{#2}\ifglxtrininsertinside#3\fi}%
5376     \ifglxtrininsertinside\else#3\fi
5377   }%
5378   }%
5379   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5380 }%
5381 \glspostlinkhook
5382 }

```

\glxtrlong

```

5383 \newrobustcmd*{\glxtrlong}{\@gl@hyp@opt\@ns@glxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
5384 \newcommand*{\ns@glxtrlong}[2] [] {%
5385   \new@ifnextchar[{\@glxtrlong{#1}{#2}}{\@glxtrlong{#1}{#2} []}%
5386 }

    Read in the final optional argument:
5387 \def\@glxtrlong#1#2[#3]{%
5388   \glsdoifexists{#2}%
5389   {%
5390     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5391     \let\glxtrifwasfirstuse\@secondoftwo
5392     \let\glsifplural\@secondoftwo
5393     \let\glscapscase\@firstofthree
5394     \let\glsinsert\@empty
5395     \def\glscustomtext{%
5396       \glsfont{\glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
5397       \ifglxtrininsertinside\else#3\fi
5398     }%
5399     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5400   }%
5401   \glspostlinkhook
5402 }

```

\Glsxtrlong

```

5403 \newrobustcmd*{\Glsxtrlong}{\@gl@hyp@opt\@ns@Glsxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
5404 \newcommand*{\ns@Glsxtrlong}[2] [] {%

```

```

5405 \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
5406 }

```

Read in the final optional argument:

```

5407 \def\@Glsxtrlong#1#2[#3]{%
5408   \glsdoifexists{#2}%
5409   {%
5410     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5411     \let\glxtrifwasfirstuse\@secondoftwo
5412     \let\gl@sifplural\@secondoftwo
5413     \let\glscapscase\@secondofthree
5414     \let\gl$insert\@empty
5415     \def\glscustomtext{%
5416       \glslongfont{\Glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
5417       \ifglxtrininsertinside\else#3\fi
5418     }%
5419     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5420   }%
5421   \glspostlinkhook
5422 }

```

\Glsxtrlong

```

5423 \newrobustcmd*{\Glsxtrlong}{\@gl@hyp@opt\@ns@Glsxtrlong}

```

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

```

5424 \newcommand*{\ns@Glsxtrlong}[2] [] {%
5425   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
5426 }

```

Read in the final optional argument:

```

5427 \def\@Glsxtrlong#1#2[#3]{%
5428   \glsdoifexists{#2}%
5429   {%
5430     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5431     \let\glxtrifwasfirstuse\@secondoftwo
5432     \let\gl@sifplural\@secondoftwo
5433     \let\glscapscase\@thirdofthree
5434     \let\gl$insert\@empty
5435     \def\glscustomtext{%
5436       \mfirstucMakeUppercase
5437       {\glslongfont{\Glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
5438       \ifglxtrininsertinside\else#3\fi
5439     }%
5440     }%
5441     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5442   }%
5443   \glspostlinkhook
5444 }

```

Plural short forms:



\glstrshortpl

```
5445 \newrobustcmd*{\glstrshortpl}{\@gls@hyp@opt\ns@glstrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
5446 \newcommand*{\ns@glstrshortpl}[2] [] {%
5447   \new@ifnextchar[{\@glstrshortpl{#1}{#2}}{\@glstrshortpl{#1}{#2} []}%
5448 }

    Read in the final optional argument:
5449 \def\@glstrshortpl#1#2[#3] {%
5450   \glsdoifexists{#2}%
5451   {%
5452     \glssetabbrvfmt{\glscategory{#2}}%
5453     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5454     \let\glstrifwasfirstuse\@secondoftwo
5455     \let\glsifplural\@firstoftwo
5456     \let\glscapscase\@firstofthree
5457     \let\glsinsert\@empty
5458     \def\glscustomtext{%
5459       \glsabbrvfont{\glsaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
5460       \ifglstrinsertinside\else#3\fi
5461     }%
5462     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5463   }%
5464   \glspostlinkhook
5465 }
```

\Glsxtrshortpl

```
5466 \newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\ns@Glsxtrshortpl}
    Define the un-starred form. Need to determine if there is a final optional argument
5467 \newcommand*{\ns@Glsxtrshortpl}[2] [] {%
5468   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
5469 }

    Read in the final optional argument:
5470 \def\@Glsxtrshortpl#1#2[#3] {%
5471   \glsdoifexists{#2}%
5472   {%
5473     \glssetabbrvfmt{\glscategory{#2}}%
5474     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5475     \let\glstrifwasfirstuse\@secondoftwo
5476     \let\glsifplural\@firstoftwo
5477     \let\glscapscase\@secondofthree
5478     \let\glsinsert\@empty
5479     \def\glscustomtext{%
5480       \Glsabbrvfont{\Glsaccessshortpl{#2}\ifglstrinsertinside#3\fi}%
5481       \ifglstrinsertinside\else#3\fi
5482     }%
5483     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5484   }%
```

```

5485 \glspostlinkhook
5486 }

```

\GLSxtrshortpl

```

5487 \newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}

  Define the un-starred form. Need to determine if there is a final optional argument
5488 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
5489   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
5490 }

  Read in the final optional argument:
5491 \def\@GLSxtrshortpl#1#2[#3] {%
5492   \glsdoidexists{#2}%
5493   {%
5494     \glsetabbrvfmt{\glscategory{#2}}%
5495     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5496     \let\glxtrifwasfirstuse\@secondoftwo
5497     \let\glsifplural\@firstoftwo
5498     \let\glscapscase\@thirdofthree
5499     \let\glsinsert\@empty
5500     \def\glscustomtext{%
5501       \mfirstucMakeUppercase
5502       {\glsabbrvfont{\glssaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
5503       \ifglxtrininsertinside\else#3\fi
5504     }%
5505   }%
5506   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5507 }%
5508 \glspostlinkhook
5509 }

```

Plural long forms:

\glxtrlongpl

```

5510 \newrobustcmd*{\glxtrlongpl}{\@gls@hyp@opt\ns@glxtrlongpl}

  Define the un-starred form. Need to determine if there is a final optional argument
5511 \newcommand*{\ns@glxtrlongpl}[2] [] {%
5512   \new@ifnextchar[{\@glxtrlongpl{#1}{#2}}{\@glxtrlongpl{#1}{#2} []}%
5513 }

  Read in the final optional argument:
5514 \def\@glxtrlongpl#1#2[#3] {%
5515   \glsdoidexists{#2}%
5516   {%
5517     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5518     \let\glxtrifwasfirstuse\@secondoftwo
5519     \let\glsifplural\@firstoftwo
5520     \let\glscapscase\@firstofthree
5521     \let\glsinsert\@empty

```

```

5522 \def\glscustomtext{%
5523 \glslongfont{\glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
5524 \ifglxtrinsertinside\else#3\fi
5525 }%
5526 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5527 }%
5528 \glspostlinkhook
5529 }

```

\Glsxtrlongpl

```

5530 \newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\@ns@Glsxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
5531 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
5532 \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
5533 }
    Read in the final optional argument:
5534 \def\@Glsxtrlongpl#1#2[#3]{%
5535 \glsdofexists{#2}%
5536 {%
5537 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5538 \let\glxtrifwasfirstuse\@secondoftwo
5539 \let\glsifplural\@firstoftwo
5540 \let\glscapscase\@secondofthree
5541 \let\glsinsert\@empty
5542 \def\glscustomtext{%
5543 \glslongfont{\Glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
5544 \ifglxtrinsertinside\else#3\fi
5545 }%
5546 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5547 }%
5548 \glspostlinkhook
5549 }

```

\GLSxtrlongpl

```

5550 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
5551 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
5552 \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
5553 }
    Read in the final optional argument:
5554 \def\@GLSxtrlongpl#1#2[#3]{%
5555 \glsdofexists{#2}%
5556 {%
5557 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5558 \let\glxtrifwasfirstuse\@secondoftwo
5559 \let\glsifplural\@firstoftwo
5560 \let\glscapscase\@thirdofthree

```

```

5561 \let\glsinsert\@empty
5562 \def\glscustomtext{%
5563     \mfirstucMakeUppercase
5564     {\glslongfont{\glsaccesslongpl{#2}\ifglxtrinsertinside#3\fi}%
5565     \ifglxtrinsertinside\else#3\fi
5566     }%
5567 }%
5568 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5569 }%
5570 \glspostlinkhook
5571 }

```

`\glssetabbrvfmt` Set the current format for the given category (or the abbreviation category if unset).

```

5572 \newcommand*{\glssetabbrvfmt}[1]{%
5573     \ifcsdef{@glsabbrv@current@#1}%
5574     {\glxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
5575     {\glxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
5576 }

```

`sxtrgenabbrvfmt` Similar to `\glsngenacfmt`, but for abbreviations.

```

5577 \newcommand*{\glxtrgenabbrvfmt}{%
5578     \ifdefempty\glscustomtext
5579     {%
5580         \ifglused\glslabel
5581         {%

```

Subsequent use:

```

5582         \glsifplural
5583         {%

```

Subsequent plural form:

```

5584         \glscapscase
5585         {%

```

Subsequent plural form, don't adjust case:

```

5586         \glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert
5587         }%
5588         {%

```

Subsequent plural form, make first letter upper case:

```

5589         \glsabbrvfont{\Glsaccessshortpl{\glslabel}}\glsinsert
5590         }%
5591         {%

```

Subsequent plural form, all caps:

```

5592         \mfirstucMakeUppercase
5593         {\glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert}%
5594         }%
5595         }%
5596         {%

```

#### Subsequent singular form

5597        `\glscapscase`  
5598        `{%`

#### Subsequent singular form, don't adjust case:

5599        `\glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert`  
5600        `}%`  
5601        `{%`

#### Subsequent singular form, make first letter upper case:

5602        `\glsabbrvfont{\Glsaccessshort{\glslabel}}\glsinsert`  
5603        `}%`  
5604        `{%`

#### Subsequent singular form, all caps:

5605        `\mfirstucMakeUppercase`  
5606        `{\glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert}%`  
5607        `}%`  
5608        `}%`  
5609        `}%`  
5610        `{%`

#### First use:

5611        `\glsifplural`  
5612        `{%`

#### First use plural form:

5613        `\glscapscase`  
5614        `{%`

#### First use plural form, don't adjust case:

5615        `\glsxtrfullplformat{\glslabel}{\glsinsert}%`  
5616        `}%`  
5617        `{%`

#### First use plural form, make first letter upper case:

5618        `\Glsxtrfullplformat{\glslabel}{\glsinsert}%`  
5619        `}%`  
5620        `{%`

#### First use plural form, all caps:

5621        `\mfirstucMakeUppercase`  
5622        `{\glsxtrfullplformat{\glslabel}{\glsinsert}}%`  
5623        `}%`  
5624        `}%`  
5625        `{%`

#### First use singular form

5626        `\glscapscase`  
5627        `{%`

#### First use singular form, don't adjust case:

5628        `\glsxtrfullformat{\glslabel}{\glsinsert}%`

```

5629     }%
5630     {%

```

First use singular form, make first letter upper case:

```

5631         \Glsxtrfullformat{\glslabel}{\glsinsert}%
5632     }%
5633     {%

```

First use singular form, all caps:

```

5634         \mfirstucMakeUppercase
5635         {\glsxtrfullformat{\glslabel}{\glsinsert}}%
5636     }%
5637 }%
5638 }%
5639 }%
5640 {%

```

User supplied text.

```

5641     \glscustomtext
5642 }%
5643 }

```

### 1.6.1 Abbreviation Styles Setup

breivationstyle

```

5644 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%
5645     \ifcsundef{@glsabbrv@dispstyle@setup@#2}%
5646     {%
5647         \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
5648     }%
5649     {%

```

Have abbreviations already been defined for this category?

```

5650     \ifcsstring{@glsabbrv@current@#1}{#2}%
5651     {%

```

Style already set.

```

5652     }%
5653     {%
5654         \def\@glsxtr@dostylewarn{%
5655             \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
5656             {%
5657                 \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
5658                     style has been switched \MessageBreak
5659                     for category ‘#1’, \MessageBreak
5660                     but there have already been entries \MessageBreak
5661                     defined for this category. Unwanted \MessageBreak
5662                     side-effects may result}}%
5663                 \@endfortrue
5664             }%
5665             \@glsxtr@dostylewarn

```

Set up the style for the given category.

```
5666      \csdef{@glsabbrv@current@#1}{#2}%
5667      \glsxtr@applyabbrvstyle{#2}%
5668      }%
5669      }%
5670 }
```

`\applyabbrvstyle` Apply the abbreviation style without existence check.

```
5671 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
5672   \csuse{@glsabbrv@dispstyle@setup@#1}%
5673   \csuse{@glsabbrv@dispstyle@fmts@#1}%
5674 }
```

`\r@applyabbrvfmt` Only apply the style formats.

```
5675 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
5676   \csuse{@glsabbrv@dispstyle@fmts@#1}%
5677 }
```

`\newabbreviationstyle` This is different from `\newacronymstyle`. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```
5678 \newcommand*{\newabbreviationstyle}[3]{%
5679   \ifcsdef{@glsabbrv@dispstyle@setup@#1}
5680   {%
5681     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
5682       defined}{}%
5683   }%
5684   {%
5685     \csdef{@glsabbrv@dispstyle@setup@#1}{%

```

Initialise hook to do nothing. The style may change this.

```
5686     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
5687     #2}%
5688     \csdef{@glsabbrv@dispstyle@fmts@#1}{%

```

Assume in-line form is the same as first use. The style may change this.

```
5689     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
5690     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
5691     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
5692     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
5693     #3}%
5694   }%
5695 }
```

`\newabbreviationstyle`

```
5696 \newcommand*{\renewabbreviationstyle}[3]{%
5697   \ifcsundef{@glsabbrv@dispstyle@setup@#1}
5698   {%
5699     \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%

```

```

5700 }%
5701 {%
5702   \csdef{@glsabbrv@dispstyle@setup@#1}{%
      Initialise hook to do nothing. The style may change this.
5703     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
5704     #2}%
5705   \csdef{@glsabbrv@dispstyle@fmts@#1}{%
      Assume in-line form is the same as first use. The style may change this.
5706     \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
5707     \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
5708     \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
5709     \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
5710     #3}%
5711 }%
5712 }

```

**abbreviationstyle** Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```

5713 \newcommand*{\letabbreviationstyle}[2]{%
5714   \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
5715   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
5716 }

```

**deprecated@abbrstyle** `\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}`

Define a synonym for a deprecated abbreviation style.

```

5717 \newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
5718   \csdef{@glsabbrv@dispstyle@setup@#1}{%
5719     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
5720     \csuse{@glsabbrv@dispstyle@setup@#2}%
5721   }%
5722   \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
5723 }

```

**deprecatedAbbrStyle** Generate warning for deprecated style use.

```

5724 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
5725   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
5726   use ‘#2’ instead}%
5727 }

```

**eAbbrStyleSetup**

```

5728 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
5729   \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
5730   {%
5731     \PackageError{glossaries-extra}%

```



```

5732     {Unknown abbreviation style definitions ‘#1’}{}%
5733 }%
5734 {%
5735     \csname @glsabbrv@dispstyle@setup@#1\endcsname
5736 }%
5737 }

```

seAbbrStyleFmts

```

5738 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
5739     \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
5740     {%
5741         \PackageError{glossaries-extra}%
5742         {Unknown abbreviation style formats ‘#1’}{}%
5743     }%
5744     {%
5745         \csname @glsabbrv@dispstyle@fmts@#1\endcsname
5746     }%
5747 }

```

## 1.6.2 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.

```

5748 \newif\ifglsxtrinsertinside
5749 \glsxtrinsertinsidefalse

```

long-short

```

5750 \newabbreviationstyle{long-short}%
5751 {%
5752     \renewcommand*{\CustomAbbreviationFields}{%
5753         name={\protect\glsabbrvfont{\the\glsshorttok}},
5754         sort={\the\glsshorttok},
5755         first={\protect\glsfirstlongfont{\the\glslongtok}%
5756             \protect\glsxtrfullsep{\the\glslabeltok}%
5757             (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
5758         firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
5759             \protect\glsxtrfullsep{\the\glslabeltok}%
5760             (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
5761         plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
5762         description={\the\glslongtok}}%

```

Unset the regular attribute if it has been set.

```

5763 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5764   \glshasattribute{\the\glslabeltok}{regular}%
5765   {%
5766     \glissetattribute{\the\glslabeltok}{regular}{false}%
5767   }%
5768   {}%
5769 }%
5770 }%
5771 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5772 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
5773 \renewcommand*{\glssabrvfont}[1]{\glssabrvdefaultfont{##1}}%
5774 \renewcommand*{\glssfirstabbrvfont}[1]{\glssfirstabbrvdefaultfont{##1}}%
5775 \renewcommand*{\glssfirstlongfont}[1]{\glssfirstlongdefaultfont{##1}}%
5776 \renewcommand*{\glsslongfont}[1]{\glsslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

5777 \renewcommand*{\glsxtrfullformat}[2]{%
5778   \glssfirstlongfont{\glssaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
5779   \ifglsxtrininsertinside\else##2\fi
5780   \glsxtrfullsep{##1}%
5781   (\glssfirstabbrvfont{\glssaccessshort{##1}})%
5782 }%
5783 \renewcommand*{\glsxtrfullplformat}[2]{%
5784   \glssfirstlongfont{\glssaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
5785   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5786   (\glssfirstabbrvfont{\glssaccessshortpl{##1}})%
5787 }%
5788 \renewcommand*{\Glsxtrfullformat}[2]{%
5789   \glssfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%
5790   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5791   (\glssfirstabbrvfont{\glssaccessshort{##1}})%
5792 }%
5793 \renewcommand*{\Glsxtrfullplformat}[2]{%
5794   \glssfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%
5795   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5796   (\glssfirstabbrvfont{\glssaccessshortpl{##1}})%
5797 }%
5798 }

```

Set this as the default style for general abbreviations:

```

5799 \setabbreviationstyle{long-short}

```

ngshortdescsort

```

5800 \newcommand*{\glsxtrlongshortdescsort}{\the\glsslongtok\space(\the\glssshorttok)}

```

long-short-desc User supplies description. The long form is included in the name.

```

5801 \newabbreviationstyle{long-short-desc}%
5802 {%
5803   \renewcommand*{\CustomAbbreviationFields}{%
5804     name={\protect\glxtrfullformat{\the\glslabeltok}{}},
5805     sort={\glxtrlongshortdescsort},%
5806     first={\protect\glsfirstlongfont{\the\glslongtok}%
5807       \protect\glxtrfullsep{\the\glslabeltok}%
5808       (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
5809     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
5810       \protect\glxtrfullsep{\the\glslabeltok}%
5811       (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%

```

The text key should only have the short form.

```

5812   text={\protect\glsabbrvfont{\the\glsshorttok}},%
5813   plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
5814 }%

```

Unset the regular attribute if it has been set.

```

5815 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5816   \glshasattribute{\the\glslabeltok}{regular}%
5817   {%
5818     \glissetattribute{\the\glslabeltok}{regular}{false}%
5819   }%
5820   {}%
5821 }%
5822 }%
5823 {%
5824   \GlsXtrUseAbbrStyleFmts{long-short}%
5825 }

```

**short-long** Short form followed by long form in parenthesis on first use.

```

5826 \newabbreviationstyle{short-long}%
5827 {%
5828   \renewcommand*{\CustomAbbreviationFields}{%
5829     name={\protect\glsabbrvfont{\the\glsshorttok}},
5830     sort={\the\glsshorttok},
5831     description={\the\glslongtok},%
5832     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
5833       \protect\glxtrfullsep{\the\glslabeltok}%
5834       (\protect\glsfirstlongfont{\the\glslongtok})},%
5835     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
5836       \protect\glxtrfullsep{\the\glslabeltok}%
5837       (\protect\glsfirstlongfont{\the\glslongpltok})},%
5838     plural={\protect\glsabbrvfont{\the\glsshortpltok}}%

```

Unset the regular attribute if it has been set.

```

5839 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5840   \glshasattribute{\the\glslabeltok}{regular}%

```

```

5841   {%
5842     \glssetattribute{\the\glslabeltok}{regular}{false}%
5843   }%
5844   {}%
5845 }%
5846 }%
5847 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5848 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
5849 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
5850 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
5851 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
5852 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

5853 \renewcommand*\glsxtrfullformat[2]{%
5854   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
5855   \ifglsxtrininsertinside\else##2\fi
5856   \glsxtrfullsep{##1}%
5857   (\glsfirstlongfont{\glsaccesslong{##1}})%
5858 }%
5859 \renewcommand*\glsxtrfullplformat[2]{%
5860   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
5861   \ifglsxtrininsertinside\else##2\fi
5862   \glsxtrfullsep{##1}%
5863   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
5864 }%
5865 \renewcommand*\Glsxtrfullformat[2]{%
5866   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
5867   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5868   (\glsfirstlongfont{\glsaccesslong{##1}})%
5869 }%
5870 \renewcommand*\Glsxtrfullplformat[2]{%
5871   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
5872   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5873   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
5874 }%
5875 }

```

`short-long-desc` User supplies description. The long form is included in the name.

```

5876 \newabbreviationstyle{short-long-desc}%
5877 {%
5878   \renewcommand*\CustomAbbreviationFields{%
5879     name={\protect\glsxtrfullformat{\the\glslabeltok}{}},
5880     sort={\the\glsshorttok},%
5881     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
5882     \protect\glsxtrfullsep{\the\glslabeltok}%
5883     (\protect\glsfirstlongfont{\the\glslongtok}}),%
5884     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%

```

```

5885 \protect\glstrfullsep{\the\glslabeltok}%
5886 (\protect\glslongfont{\the\glslongpltok}}),%
5887 text={\protect\glssabrvfont{\the\glsshorttok}},%
5888 plural={\protect\glssabrvfont{\the\glsshortpltok}}%
5889 }%
    Unset the regular attribute if it has been set.
5890 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5891 \glshasattribute{\the\glslabeltok}{regular}%
5892 {%
5893 \glsssetAttribute{\the\glslabeltok}{regular}{false}%
5894 }%
5895 {}%
5896 }%
5897 }%
5898 {%
5899 \GlsXtrUseAbbrStyleFmts{short-long}%
5900 }

```

`\glslongfootnotefont` Only used by the “footnote” styles.

```
5901 \newcommand*{\glslongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

`\glslongfootnotefont` Only used by the “footnote” styles.

```
5902 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%
```

`\glsxtrabrvfootnote` `\glsxtrabrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument `<long>` includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```
5903 \newcommand*{\glsxtrabrvfootnote}[2]{\footnote{#2}}
```

`\footnote` Short form followed by long form in footnote on first use.

```

5904 \newabbreviationstyle{footnote}%
5905 {%
5906 \renewcommand*{\CustomAbbreviationFields}{%
5907 name={\protect\glssabrvfont{\the\glsshorttok}},
5908 sort={\the\glsshorttok},
5909 description={\the\glslongtok},%
5910 first={\protect\glslongfont{\the\glslongpltok}}%
5911 \protect\glsxtrabrvfootnote{\the\glslabeltok}%
5912 {\protect\glslongfont{\the\glslongpltok}}},%

```

```

5913 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
5914 \protect\glsextrabbrvfootnote{\the\glslabeltok}%
5915 {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%

```

```

5916 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.

```

5917 \renewcommand*\GlsXtrPostNewAbbreviation{%
5918 \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
5919 \glsattribute{\the\glslabeltok}{regular}%
5920 {%
5921 \glssetattribute{\the\glslabeltok}{regular}{false}%
5922 }%
5923 {}%
5924 }%
5925}%
5926{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5927 \renewcommand*\abbrvpluralsuffix{\glsextrabbrvpluralsuffix}%
5928 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
5929 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
5930 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongfootnotefont{##1}}%
5931 \renewcommand*\glslongfont[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```

5932 \renewcommand*\glsextrfullformat[2]{%
5933 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsextrininsertinside##2\fi}%
5934 \ifglsextrininsertinside\else##2\fi
5935 \protect\glsextrabbrvfootnote{##1}%
5936 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
5937}%
5938 \renewcommand*\glsextrfullplformat[2]{%
5939 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsextrininsertinside##2\fi}%
5940 \ifglsextrininsertinside\else##2\fi
5941 \protect\glsextrabbrvfootnote{##1}%
5942 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
5943}%
5944 \renewcommand*\Glsxtrfullformat[2]{%
5945 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsextrininsertinside##2\fi}%
5946 \ifglsextrininsertinside\else##2\fi
5947 \protect\glsextrabbrvfootnote{##1}%
5948 {\glsfirstlongfootnotefont{\Glsaccesslong{##1}}}%
5949}%
5950 \renewcommand*\Glsxtrfullplformat[2]{%
5951 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsextrininsertinside##2\fi}%
5952 \ifglsextrininsertinside\else##2\fi
5953 \protect\glsextrabbrvfootnote{##1}%
5954 {\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}}}%
5955}%

```

The first use full form and the inline full form use the short (long) style.

```

5956 \renewcommand*{\glxtrinlinefullformat}[2]{%
5957   \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5958   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5959   (\glsfirstlongfootnotefont{\glssaccesslong{##1}})%
5960 }%
5961 \renewcommand*{\glxtrinlinefullplformat}[2]{%
5962   \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5963   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5964   (\glsfirstlongfootnotefont{\glssaccesslongpl{##1}})%
5965 }%
5966 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
5967   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5968   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5969   (\glsfirstlongfootnotefont{\Glsaccesslong{##1}})%
5970 }%
5971 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
5972   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5973   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5974   (\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}})%
5975 }%
5976 }

```

short-footnote

```

5977 \letabbreviationstyle{short-footnote}{footnote}

```

postfootnote Similar to the above but the footnote is placed afterwards, outside the link. This avoids nested links and can also move the footnote marker after any following punctuation mark. Pre v1.07 included `\footnote` in the first keys, which was incorrect as it caused duplicate footnotes.

```

5978 \newabbreviationstyle{postfootnote}%
5979 {%
5980   \renewcommand*{\CustomAbbreviationFields}{%
5981     name={\protect\glsabbrvfont{\the\glssshorttok}},
5982     sort={\the\glssshorttok},
5983     description={\the\glslongtok},%
5984     first={\protect\glsfirstabbrvfont{\the\glssshorttok}},%
5985     firstplural={\protect\glsfirstabbrvfont{\the\glssshortpltok}},%
5986     plural={\protect\glsabbrvfont{\the\glssshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

5987 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5988   \csdef{glxtrpostlink\glscategorylabel}{%
5989     \glxtrifwasfirstuse
5990     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

5991      \glxtrdopostpunc{\protect\glxtrabbrvfootnote{\glslabel}%
5992      {\glslfirstlongfootnotefont{\glslentrylong{\glslabel}}}}%
5993      }%
5994      {}%
5995      }%
5996      \glshasattribute{\the\glslabeltok}{regular}%
5997      {%
5998      \glissetattribute{\the\glslabeltok}{regular}{false}%
5999      }%
6000      {}%
6001      }%

```

The footnote needs to be suppressed in the inline form, so `\glxtrfull` must set the first use switch off.

```

6002 \renewcommand*{\glxtrsetupfulldefs}{%
6003   \let\glxtrifwasfirstuse\@secondoftwo
6004 }%
6005}%
6006{%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6007 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
6008 \renewcommand*{\glslabbrvfont}[1]{\glslabbrvdefaultfont{##1}}%
6009 \renewcommand*{\glslfirstabbrvfont}[1]{\glslfirstabbrvdefaultfont{##1}}%
6010 \renewcommand*{\glslfirstlongfont}[1]{\glslfirstlongfootnotefont{##1}}%
6011 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

6012 \renewcommand*{\glxtrfullformat}[2]{%
6013   \glslfirstabbrvfont{\glslaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6014   \ifglxtrininsertinside\else##2\fi
6015 }%
6016 \renewcommand*{\glxtrfullplformat}[2]{%
6017   \glslfirstabbrvfont{\glslaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
6018   \ifglxtrininsertinside\else##2\fi
6019 }%
6020 \renewcommand*{\Glsxtrfullformat}[2]{%
6021   \glslfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6022   \ifglxtrininsertinside\else##2\fi
6023 }%
6024 \renewcommand*{\Glsxtrfullplformat}[2]{%
6025   \glslfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
6026   \ifglxtrininsertinside\else##2\fi
6027 }%

```

The first use full form and the inline full form use the short (long) style.

```

6028 \renewcommand*{\glxtrininlinefullformat}[2]{%
6029   \glslfirstabbrvfont{\glslaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6030   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
6031   (\glslfirstlongfootnotefont{\glslaccesslong{##1}})%
6032 }%

```



```

6033 \renewcommand*{\glxtrinlinefullplformat}[2]{%
6034   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6035   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6036   (\glsfirstlongfootnotefont{\glsaccesslongpl{##1}})%
6037 }%
6038 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
6039   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
6040   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6041   (\glsfirstlongfootnotefont{\Glsaccesslong{##1}})%
6042 }%
6043 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
6044   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6045   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6046   (\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}})%
6047 }%
6048 }

```

rt-postfootnote

```
6049 \letabbreviationstyle{short-postfootnote}{postfootnote}
```

**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.

```

6050 \newabbreviationstyle{short}%
6051 {%
6052   \renewcommand*{\CustomAbbreviationFields}{%
6053     name={\protect\glsabbrvfont{\the\glsshorttok}},
6054     sort={\the\glsshorttok},
6055     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
6056     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
6057     text={\protect\glsabbrvfont{\the\glsshorttok}},
6058     plural={\protect\glsabbrvfont{\the\glsshortpltok}},
6059     description={\the\glslongtok}}%
6060   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6061     \glssetattribute{\the\glslabeltok}{regular}{true}}%
6062 }%
6063 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6064 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
6065 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
6066 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
6067 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
6068 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

6069 \renewcommand*{\glxtrinlinefullformat}[2]{%
6070   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
6071   \ifglxtrinsertinside##2\fi}%

```

```

6072 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6073 (\glfirstlongfont{\glsaccesslong{##1}})%
6074 }%
6075 \renewcommand*{\glxtrinlinefullplformat}[2]{%
6076 \protect\glfirstabbrvfont{\glsaccessshortpl{##1}%
6077 \ifglxtrinsertinside##2\fi}%
6078 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6079 (\glfirstlongfont{\glsaccesslongpl{##1}})%
6080 }%
6081 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
6082 \protect\glfirstabbrvfont{\glsaccessshort{##1}%
6083 \ifglxtrinsertinside##2\fi}%
6084 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6085 (\glfirstlongfont{\Glsaccesslong{##1}})%
6086 }%
6087 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
6088 \protect\glfirstabbrvfont{\glsaccessshortpl{##1}%
6089 \ifglxtrinsertinside##2\fi}%
6090 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6091 (\glfirstlongfont{\Glsaccesslongpl{##1}})%
6092 }%

```

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.

```

6093 \renewcommand*{\glxtrfullformat}[2]{%
6094 \glfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
6095 \ifglxtrinsertinside\else##2\fi
6096 }%
6097 \renewcommand*{\glxtrfullplformat}[2]{%
6098 \glfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6099 \ifglxtrinsertinside\else##2\fi
6100 }%
6101 \renewcommand*{\Glsxtrfullformat}[2]{%
6102 \glfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
6103 \ifglxtrinsertinside\else##2\fi
6104 }%
6105 \renewcommand*{\Glsxtrfullplformat}[2]{%
6106 \glfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6107 \ifglxtrinsertinside\else##2\fi
6108 }%
6109 }

```

Set this as the default style for acronyms:

```

6110 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

6111 \letabbreviationstyle{short-nolong}{short}

```

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.

```

6112 \newabbreviationstyle{short-desc}%
6113 {%
6114   \renewcommand*{\CustomAbbreviationFields}{%
6115     name={\protect\glxtrinlinefullformat{\the\glslabeltok}{}},
6116     sort={\the\glsshorttok},
6117     first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
6118     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
6119     text={\protect\glssabbvfont{\the\glsshorttok}},
6120     plural={\protect\glssabbvfont{\the\glsshortpltok}},
6121     description={\the\glslongtok}}%
6122   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6123     \glsssetAttribute{\the\glslabeltok}{regular}{true}}%
6124 }%
6125 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6126 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
6127 \renewcommand*{\glssabbvfont}[1]{\glssabbvdefaultfont{##1}}%
6128 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
6129 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
6130 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

6131 \renewcommand*{\glxtrinlinefullformat}[2]{%
6132   \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6133   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
6134   (\glsfirstlongfont{\glssaccesslong{##1}})%
6135 }%
6136 \renewcommand*{\glxtrinlinefullplformat}[2]{%
6137   \glsfirstabbrvfont{\glssaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
6138   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
6139   (\glsfirstlongfont{\glssaccesslongpl{##1}})%
6140 }%
6141 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
6142   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6143   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
6144   (\glsfirstlongfont{\Glsaccesslong{##1}})%
6145 }%
6146 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
6147   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
6148   \ifglxtrininsertinside\else##2\fi\glxtrfullsep{##1}%
6149   (\glsfirstlongfont{\Glsaccesslongpl{##1}})%
6150 }%

```

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.

```

6151 \renewcommand*{\glxtrfullformat}[2]{%
6152   \glsfirstabbrvfont{\glssaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6153   \ifglxtrininsertinside\else##2\fi
6154 }%
6155 \renewcommand*{\glxtrfullplformat}[2]{%

```

```

6156 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6157 \ifglxtrinsertinside\else##2\fi
6158 }%
6159 \renewcommand*{\Glsxtrfullformat}[2]{%
6160 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
6161 \ifglxtrinsertinside\else##2\fi
6162 }%
6163 \renewcommand*{\Glsxtrfullplformat}[2]{%
6164 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6165 \ifglxtrinsertinside\else##2\fi
6166 }%
6167 }

```

ort-nolong-desc

```
6168 \letabbreviationstyle{short-nolong-desc}{short-desc}
```

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.

```

6169 \newabbreviationstyle{long-desc}%
6170 {%
6171 \renewcommand*{\CustomAbbreviationFields}{%
6172 name={\protect\protect\glsfirstlongfont{\the\glslongtok}},
6173 sort={\the\glslongtok},
6174 first={\protect\glsfirstlongfont{\the\glslongtok}},
6175 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
6176 text={\the\glslongtok},
6177 plural={\the\glslongpltok}}%
6178 }%
6179 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6180 \glssetattribute{\the\glslabeltok}{regular}{true}}%
6181 }%
6182 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6183 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
6184 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
6185 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
6186 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
6187 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the long format followed by the short form in parentheses.

```

6188 \renewcommand*{\glxtrinlinefullformat}[2]{%
6189 \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6190 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6191 (\protect\glsfirstabbrvfont{\glsaccessshort{##1}})%
6192 }%
6193 \renewcommand*{\glxtrinlinefullplformat}[2]{%
6194 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6195 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%

```

```

6196 (\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
6197 }%
6198 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
6199 \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6200 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6201 (\protect\glsfirstabbrvfont{\glsaccessshort{##1}})%
6202 }%
6203 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
6204 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6205 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
6206 (\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
6207 }%

```

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.

```

6208 \renewcommand*{\glxtrfullformat}[2]{%
6209 \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6210 \ifglxtrinsertinside\else##2\fi
6211 }%
6212 \renewcommand*{\glxtrfullplformat}[2]{%
6213 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6214 \ifglxtrinsertinside\else##2\fi
6215 }%
6216 \renewcommand*{\Glsxtrfullformat}[2]{%
6217 \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6218 \ifglxtrinsertinside\else##2\fi
6219 }%
6220 \renewcommand*{\Glsxtrfullplformat}[2]{%
6221 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6222 \ifglxtrinsertinside\else##2\fi
6223 }%
6224 }

```

ng-noshort-desc Provide a synonym that matches similar styles.

```

6225 \letabbreviationstyle{long-noshort-desc}{long-desc}

```

long It doesn't really make a great deal of sense to have a long-only style that doesn't have a description, but the best course of action here is to use the short form as the name and the long form as the description.

```

6226 \newabbreviationstyle{long}%
6227 {%
6228 \renewcommand*{\CustomAbbreviationFields}{%
6229 name={\protect\glsabbrvfont{\the\glsshorttok}},
6230 sort={\the\glsshorttok},
6231 first={\protect\glsfirstlongfont{\the\glslongtok}},
6232 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
6233 text={\the\glslongtok},
6234 plural={\the\glslongpltok},%
6235 description={\the\glslongtok}%

```

```

6236 }%
6237 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6238   \glsssetattribute{\the\glslabeltok}{regular}{true}}%
6239 }%
6240 {%
6241   \GlsXtrUseAbbrStyleFmts{long-desc}%
6242 }

```

long-noshort Provide a synonym that matches similar styles.

```
6243 \letabbreviationstyle{long-noshort}{long}
```

### 1.6.3 Predefined Styles (Small Capitals)

These styles use:

\glsxtrscfont

```
6244 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

sxtrfirstscfont

```
6245 \newcommand*{\glsxtrfirstscfont}[1]{\glsxtrscfont{#1}}
```

and for the default short form suffix:

\glsxtrscsuffix

```
6246 \newcommand*{\glsxtrscsuffix}{\glstextup{\glxtrabbrvpluralsuffix}}
```

long-short-sc

```

6247 \newabbreviationstyle{long-short-sc}%
6248 {%
6249   \GlsXtrUseAbbrStyleSetup{long-short}%
6250 }%
6251 {%

```

Mostly as long-short style:

```
6252 \GlsXtrUseAbbrStyleFmts{long-short}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```

6253 \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
6254 \renewcommand*{\glssabrvfont}[1]{\glsxtrscfont{##1}}%
6255 \renewcommand*{\glssfirstabrvfont}[1]{\glsxtrfirstscfont{##1}}%
6256 }

```

g-short-sc-desc

```

6257 \newabbreviationstyle{long-short-sc-desc}%
6258 {%
6259   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
6260 }%
6261 {%

```

Mostly as long-short-desc style:

```
6262 \GlsXtrUseAbbrStyleFmts{long-short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6263 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6264 \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
6265 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
6266 }
```

Now the short (long) version

```
6267 \newabbreviationstyle{short-sc-long}%
6268 {%
6269 \GlsXtrUseAbbrStyleSetup{short-long}%
6270 }%
6271 {%
```

Mostly as short-long style:

```
6272 \GlsXtrUseAbbrStyleFmts{short-long}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6273 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6274 \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
6275 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
6276 }
```

As before but user provides description

```
6277 \newabbreviationstyle{short-sc-long-desc}%
6278 {%
6279 \GlsXtrUseAbbrStyleSetup{short-long-desc}%
6280 }%
6281 {%
```

Mostly as short-long-desc style:

```
6282 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6283 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6284 \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
6285 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
6286 }
```

short-sc

```
6287 \newabbreviationstyle{short-sc}%
6288 {%
6289 \GlsXtrUseAbbrStyleSetup{short-nolong}%
6290 }%
6291 {%
```

Mostly as short style:

```
6292 \GlsXtrUseAbbrStyleFmts{short-nolong}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6293 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6294 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
6295 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
6296 }
```

short-sc-nolong

```
6297 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
6298 \newabbreviationstyle{short-sc-desc}%
6299 {%
6300 \GlsXtrUseAbbrStyleSetup{short-desc}%
6301 }%
6302 {%
```

Mostly as short style:

```
6303 \GlsXtrUseAbbrStyleFmts{short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6304 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6305 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
6306 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
6307 }
```

-sc-nolong-desc

```
6308 \letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```
6309 \newabbreviationstyle{long-noshort-sc}%
6310 {%
6311 \GlsXtrUseAbbrStyleSetup{long-noshort}%
6312 }%
6313 {%
```

Mostly as long style:

```
6314 \GlsXtrUseAbbrStyleFmts{long-noshort}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6315 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6316 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
6317 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
6318 }
```

long-sc Backward compatibility:

```
6319 \@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.

```
6320 \newabbreviationstyle{long-noshort-sc-desc}%
6321 {%
6322   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
6323 }%
6324 {%
```

Mostly as long style:

```
6325   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6326   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6327   \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
6328   \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
6329 }
```

long-desc-sc    Backward compatibility:

```
6330 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

short-sc-footnote

```
6331 \newabbreviationstyle{short-sc-footnote}%
6332 {%
6333   \GlsXtrUseAbbrStyleSetup{short-footnote}%
6334 }%
6335 {%
```

Mostly as long style:

```
6336   \GlsXtrUseAbbrStyleFmts{short-footnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6337   \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6338   \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
6339   \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
6340 }
```

footnote-sc    Backward compatibility:

```
6341 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}
```

sc-postfootnote

```
6342 \newabbreviationstyle{short-sc-postfootnote}%
6343 {%
6344   \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
6345 }%
6346 {%
```

Mostly as long style:

```
6347   \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6348 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6349 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
6350 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
6351 }
```

postfootnote-sc Backward compatibility:

```
6352 \@glxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}
```

## 1.6.4 Predefined Styles (Fake Small Capitals)

These styles require the relsize package, which must be loaded by the user. These styles all use:

\glxtrsmfont

```
6353 \newcommand*{\glxtrsmfont}[1]{\textsmaller{#1}}
```

glxtrfirstsmfont

```
6354 \newcommand*{\glxtrfirstsmfont}[1]{\glxtrsmfont{#1}}
```

and for the default short form suffix:

\glxtrsmsuffix

```
6355 \newcommand*{\glxtrsmsuffix}{\glxtrabbrvpluralsuffix}
```

long-short-sm

```
6356 \newabbreviationstyle{long-short-sm}%
6357 {%
6358   \GlsXtrUseAbbrStyleSetup{long-short}%
6359 }%
6360 {%
```

Mostly as long-short style:

```
6361 \GlsXtrUseAbbrStyleFmts{long-short}%
6362 \renewcommand*{\glssabrvfont[1]{\glxtrsmfont{##1}}}%
6363 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstsmfont{##1}}}%
6364 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
6365 }
```

g-short-sm-desc

```
6366 \newabbreviationstyle{long-short-sm-desc}%
6367 {%
6368   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
6369 }%
6370 {%
```

Mostly as long-short-desc style:

```
6371 \GlsXtrUseAbbrStyleFmts{long-short-desc}%
6372 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6373 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6374 \renewcommand*\{abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
6375 }
```

short-sm-long Now the short (long) version

```
6376 \newabbreviationstyle{short-sm-long}%
6377 {%
6378 \GlsXtrUseAbbrStyleSetup{short-long}%
6379 }%
6380 {%
```

Mostly as short-long style:

```
6381 \GlsXtrUseAbbrStyleFmts{short-long}%
6382 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6383 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6384 \renewcommand*\{abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
6385 }
```

rt-sm-long-desc As before but user provides description

```
6386 \newabbreviationstyle{short-sm-long-desc}%
6387 {%
6388 \GlsXtrUseAbbrStyleSetup{short-long-desc}%
6389 }%
6390 {%
```

Mostly as short-long-desc style:

```
6391 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
6392 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6393 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6394 \renewcommand*\{abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
6395 }
```

short-sm

```
6396 \newabbreviationstyle{short-sm}%
6397 {%
6398 \GlsXtrUseAbbrStyleSetup{short-nolong}%
6399 }%
6400 {%
```

Mostly as short style:

```
6401 \GlsXtrUseAbbrStyleFmts{short-nolong}%
6402 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6403 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6404 \renewcommand*\{abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
6405 }
```

short-sm-nolong

```
6406 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

```

short-sm-desc
6407 \newabbreviationstyle{short-sm-desc}%
6408 {%
6409   \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
6410 }%
6411 {%
    Mostly as short style:
6412   \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
6413   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%
6414   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%
6415   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%
6416 }

-sm-nolong-desc
6417 \letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}

long-noshort-sm  The smallcaps font will only be used if the short form is explicitly invoked through commands
                  like \glsshort.
6418 \newabbreviationstyle{long-noshort-sm}%
6419 {%
6420   \GlsXtrUseAbbrStyleSetup{long-noshort}%
6421 }%
6422 {%
    Mostly as long style:
6423   \GlsXtrUseAbbrStyleFmts{long-noshort}%
6424   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%
6425   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%
6426   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%
6427 }

long-sm  Backward compatibility:
6428 \@glstr@deprecated@abbrstyle{long-sm}{long-noshort-sm}

noshort-sm-desc  The smaller font will only be used if the short form is explicitly invoked through commands
                  like \glsshort.
6429 \newabbreviationstyle{long-noshort-sm-desc}%
6430 {%
6431   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
6432 }%
6433 {%
    Mostly as long style:
6434   \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
6435   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%
6436   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%
6437   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%
6438 }

```

long-desc-sm    Backward compatibility:  
6439 \@glsxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

short-sm-footnote

6440 \newabbreviationstyle{short-sm-footnote}%  
6441 {%  
6442    \GlsXtrUseAbbrStyleSetup{short-footnote}%  
6443 }%  
6444 {%

    Mostly as long style:

6445    \GlsXtrUseAbbrStyleFmts{short-footnote}%  
6446    \renewcommand\*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%  
6447    \renewcommand\*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%  
6448    \renewcommand\*\{abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%  
6449 }

footnote-sm    Backward compatibility:  
6450 \@glsxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

sm-postfootnote

6451 \newabbreviationstyle{short-sm-postfootnote}%  
6452 {%  
6453    \GlsXtrUseAbbrStyleSetup{short-postfootnote}%  
6454 }%  
6455 {%

    Mostly as long style:

6456    \GlsXtrUseAbbrStyleFmts{short-postfootnote}%  
6457    \renewcommand\*\glsabbrvfont[1]{\glsxtrsmfont{##1}}%  
6458    \renewcommand\*\glsfirstabbrvfont[1]{\glsxtrfirstsmfont{##1}}%  
6459    \renewcommand\*\{abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%  
6460 }

postfootnote-sm    Backward compatibility:  
6461 \@glsxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

## 1.6.5 Predefined Styles (Emphasized)

These styles use \emph for the short form.

\glsabbrvemfont

6462 \newcommand\*\{glsabbrvemfont}[1]{\emph{##1}}%

firstabbrvemfont

6463 \newcommand\*\{glsfirstabbrvemfont}[1]{\glsabbrvemfont{##1}}%

firstlongemfont    Only used by the “long-em” styles.

6464 \newcommand\*\{glsfirstlongemfont}[1]{\glsfirstlongemfont{##1}}%

`\glslongemfont` Only used by the “long-em” styles.

```
6465 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

long-short-em

```
6466 \newabbreviationstyle{long-short-em}%
6467 {%
6468   \GlsXtrUseAbbrStyleSetup{long-short}%
6469 }%
6470 {%
```

Mostly as long-short style:

```
6471   \GlsXtrUseAbbrStyleFmts{long-short}%
6472   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6473 }
```

g-short-em-desc

```
6474 \newabbreviationstyle{long-short-em-desc}%
6475 {%
6476   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
6477 }%
6478 {%
```

Mostly as long-short-desc style:

```
6479   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
6480   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6481 }
```

ong-em-short-em

```
6482 \newabbreviationstyle{long-em-short-em}%
6483 {%
```

`\glslongemfont` is used in the description since `\glsdesc` doesn't set the style.

```
6484   \renewcommand*{\CustomAbbreviationFields}{%
6485     name={\protect\glsabbrvfont{\the\glsshorttok}},
6486     sort={\the\glsshorttok},
6487     first={\protect\glsfirstlongfont{\the\glslongtok}%
6488       \protect\glxtrfullsep{\the\glslabeltok}%
6489       (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
6490     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
6491       \protect\glxtrfullsep{\the\glslabeltok}%
6492       (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
6493     plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
6494     description={\protect\glslongemfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
6495   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6496     \glshasattribute{\the\glslabeltok}{regular}%
6497     {%
6498       \glssetattribute{\the\glslabeltok}{regular}{false}%
```

```

6499     }%
6500     {}%
6501 }%
6502 }%
6503 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6504 \GlsXtrUseAbbrStyleFmts{long-short}%
6505 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6506 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
6507 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
6508 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
6509 }

```

m-short-em-desc

```

6510 \newabbreviationstyle{long-em-short-em-desc}%
6511 {%
6512 \GlsXtrUseAbbrStyleSetup{long-short-desc}%
6513 }%
6514 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6515 \GlsXtrUseAbbrStyleFmts{long-short-desc}%
6516 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6517 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
6518 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
6519 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
6520 }

```

short-em-long Now the short (long) version

```

6521 \newabbreviationstyle{short-em-long}%
6522 {%
6523 \GlsXtrUseAbbrStyleSetup{short-long}%
6524 }%
6525 {%

```

Mostly as short-long style:

```

6526 \GlsXtrUseAbbrStyleFmts{short-long}%
6527 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6528 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
6529 }

```

rt-em-long-desc As before but user provides description

```

6530 \newabbreviationstyle{short-em-long-desc}%
6531 {%
6532 \GlsXtrUseAbbrStyleSetup{short-long-desc}%
6533 }%
6534 {%

```

Mostly as short-long-desc style:

```

6535 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
6536 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6537 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6538 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
6539 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
6540 }

```

hort-em-long-em

```

6541 \newabbreviationstyle{short-em-long-em}%
6542 {%
    \glslongemfont is used in the description since \glsdesc doesn't set the style.
6543 \renewcommand*\CustomAbbreviationFields{%
6544     name={\protect\glsabbrvfont{\the\glsshorttok}},
6545     sort={\the\glsshorttok},
6546     description={\protect\glslongemfont{\the\glslongtok}},%
6547     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
6548     \protect\glstrfullsep{\the\glslabeltok}%
6549     (\protect\glsfirstlongfont{\the\glslongtok}}),%
6550     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
6551     \protect\glstrfullsep{\the\glslabeltok}%
6552     (\protect\glsfirstlongfont{\the\glslongpltok}}),%
6553     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

6554 \renewcommand*\GlsXtrPostNewAbbreviation{%
6555     \glshasattribute{\the\glslabeltok}{regular}%
6556     {%
6557         \glissetattribute{\the\glslabeltok}{regular}{false}%
6558     }%
6559     {}%
6560 }%
6561 }%
6562 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6563 \GlsXtrUseAbbrStyleFmts{short-long}%
6564 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6565 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6566 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
6567 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
6568 }

```

em-long-em-desc

```

6569 \newabbreviationstyle{short-em-long-em-desc}%
6570 {%
6571     \GlsXtrUseAbbrStyleSetup{short-long-desc}%
6572 }%
6573 {%

```



In case the user wants to mix and match font styles, these are redefined here.

```
6574 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
6575 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6576 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6577 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
6578 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
6579 }
```

short-em

```
6580 \newabbreviationstyle{short-em}%
6581 {%
6582 \GlsXtrUseAbbrStyleSetup{short-nolong}%
6583 }%
6584 {%
```

Mostly as short style:

```
6585 \GlsXtrUseAbbrStyleFmts{short-nolong}%
6586 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6587 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6588 }
```

short-em-nolong

```
6589 \letabbreviationstyle{short-em-nolong}{short-em}
```

short-em-desc

```
6590 \newabbreviationstyle{short-em-desc}%
6591 {%
6592 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
6593 }%
6594 {%
```

Mostly as short style:

```
6595 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
6596 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6597 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6598 }
```

-em-nolong-desc

```
6599 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}
```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```
6600 \newabbreviationstyle{long-noshort-em}%
6601 {%
6602 \GlsXtrUseAbbrStyleSetup{long-noshort}%
6603 }%
6604 {%
```

Mostly as long-noshort style:

```
6605 \GlsXtrUseAbbrStyleFmts{long-noshort}%
```

```

6606 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6607 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6608 }

```

long-em    Backward compatibility:

```

6609 \@glsxtr@deprecated@abbrstyle{long-em}{long-noshort-em}

```

g-em-noshort-em    The short form is explicitly invoked through commands like \glsshort.

```

6610 \newabbreviationstyle{long-em-noshort-em}%
6611 {%
6612   \renewcommand*\CustomAbbreviationFields{%
6613     name={\protect\glsabbrvfont{\the\glsshorttok}},
6614     sort={\the\glsshorttok},
6615     first={\protect\glsfirstlongfont{\the\glslongtok}},
6616     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
6617     text={\the\glslongtok},
6618     plural={\the\glslongpltok},%
6619     description={\protect\glslongemfont{\the\glslongtok}}%
6620   }%
6621   \renewcommand*\GlsXtrPostNewAbbreviation{%
6622     \glssetattribute{\the\glslabeltok}{regular}{true}}%
6623 }%
6624 {%

```

Mostly as long-noshort style:

```

6625 \GlsXtrUseAbbrStyleFmts{long-noshort}%
6626 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6627 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6628 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
6629 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
6630 }

```

noshort-em-desc    The emphasized font will only be used if the short form is explicitly invoked through commands like \glsshort.

```

6631 \newabbreviationstyle{long-noshort-em-desc}%
6632 {%
6633   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
6634 }%
6635 {%

```

Mostly as long style:

```

6636 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
6637 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6638 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6639 }

```

long-desc-em    Backward compatibility:

```

6640 \@glsxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}

```

noshort-em-desc The short form is explicitly invoked through commands like `\glsshort`. The long form is emphasized.

```
6641 \newabbreviationstyle{long-em-noshort-em-desc}%
6642 {%
6643   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
6644 }%
6645 {%
```

Mostly as long style:

```
6646 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
6647 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6648 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6649 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
6650 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
6651 }
```

ort-em-footnote

```
6652 \newabbreviationstyle{short-em-footnote}%
6653 {%
6654   \GlsXtrUseAbbrStyleSetup{short-footnote}%
6655 }%
6656 {%
```

Mostly as long style:

```
6657 \GlsXtrUseAbbrStyleFmts{short-footnote}%
6658 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6659 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6660 }
```

footnote-em Backward compatibility:

```
6661 \@glxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}
```

em-postfootnote

```
6662 \newabbreviationstyle{short-em-postfootnote}%
6663 {%
6664   \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
6665 }%
6666 {%
```

Mostly as long style:

```
6667 \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
6668 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6669 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6670 }
```

postfootnote-em Backward compatibility:

```
6671 \@glxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}
```

### 1.6.6 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.

```
6672 \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.

```
6673 \ifdef\glscurrentfieldvalue
6674 {
6675   \newcommand*{\glxstruserparen}[2]{%
6676     \glxstrfullsep{#2}%
6677     (#1\ifglshasfield{\glxstruserfield}{#2}{, \glscurrentfieldvalue}{})%
6678   }
6679 }
6680 {
6681   \newcommand*{\glxstruserparen}[2]{%
6682     \glxstrfullsep{#2}%
6683     (#1\ifglshasfield{\glxstruserfield}{#2}{, \@glo@thisvalue}{})%
6684   }
6685 }
```

Font used for short form:

`lsabbrvuserfont`

```
6686 \newcommand*{\glabbrvuserfont}[1]{#1}
```

Font used for short form on first use:

`stabbrvuserfont`

```
6687 \newcommand*{\glfirstabbrvuserfont}[1]{\glabbrvuserfont{#1}}
```

Font used for long form:

`glslonguserfont`

```
6688 \newcommand*{\glslonguserfont}[1]{#1}
```

Font used for long form on first use:

`rstlonguserfont`

```
6689 \newcommand*{\glfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

`lsxtrusersuffix`

```
6690 \newcommand*{\lsxtrusersuffix}{\lsxtrabbrvpluralsuffix}
```

long-short-user

```
6691 \newabbreviationstyle{long-short-user}%  
6692 {%
```

\glslonguserfont is used in the description since \glsdesc doesn't set the style.

```
6693 \renewcommand*{\CustomAbbreviationFields}{%  
6694   name={\protect\glsabbrvfont{\the\glsshorttok}},  
6695   sort={\the\glsshorttok},  
6696   first={\protect\glsfirstlongfont{\the\glslongtok}}%  
6697   \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}{\the\glslabeltok}},  
6698   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}}%  
6699   \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}{\the\glslabeltok}}%  
  
6700   plural={\protect\glsabbrvfont{\the\glsshortpltok}},%  
6701   description={\protect\glslonguserfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
6702 \renewcommand*{\GlsXtrPostNewAbbreviation}{%  
6703   \glsattribute{\the\glslabeltok}{regular}%  
6704   {%  
6705     \glssetattribute{\the\glslabeltok}{regular}{false}%  
6706   }%  
6707   {}%  
6708 }%  
6709 }%  
6710 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
6711 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%  
6712 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%  
6713 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%  
6714 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%  
6715 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
6716 \renewcommand*{\glsxtrfullformat}[2]{%  
6717   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%  
6718   \ifglsxtrininsertinside\else##2\fi  
6719   \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%  
6720 }%  
6721 \renewcommand*{\glsxtrfullplformat}[2]{%  
6722   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrininsertinside##2\fi}%  
6723   \ifglsxtrininsertinside\else##2\fi  
6724   \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%  
6725 }%  
6726 \renewcommand*{\Glsxtrfullformat}[2]{%  
6727   \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrininsertinside##2\fi}%  
6728   \ifglsxtrininsertinside\else##2\fi  
6729   \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%  
6730 }%  
6731 \renewcommand*{\Glsxtrfullplformat}[2]{%
```

```

6732 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6733 \ifglxtrinsertinside\else##2\fi
6734 \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
6735 }%
6736 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

6737 \newabbreviationstyle{long-postshort-user}%
6738 {%
6739 \renewcommand*{\CustomAbbreviationFields}{%
6740 name={\protect\glsabbrvfont{\the\glsshorttok}},
6741 sort={\the\glsshorttok},
6742 first={\protect\glsfirstlongfont{\the\glslongtok}},%
6743 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%
6744 plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
6745 description={\protect\glslonguserfont{\the\glslongtok}}}%
6746 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6747 \csdef{glxtrpostlink\glscategorylabel}{%
6748 \glxtrifwasfirstuse
6749 {%
6750 \glxtruserparen
6751 {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
6752 {\glslabel}%
6753 }%
6754 }%
6755 }%
6756 \glshasattribute{\the\glslabeltok}{regular}%
6757 {%
6758 \glissetattribute{\the\glslabeltok}{regular}{false}%
6759 }%
6760 }%
6761 }%
6762 }%
6763 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6764 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
6765 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
6766 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
6767 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
6768 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

6769 \renewcommand*{\glxtrfullformat}[2]{%
6770 \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6771 \ifglxtrinsertinside\else##2\fi
6772 }%
6773 \renewcommand*{\glxtrfullplformat}[2]{%
6774 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%

```

```

6775 \ifglxtrinsertinside\else##2\fi
6776 }%
6777 \renewcommand*{\Glsxtrfullformat}[2]{%
6778 \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6779 \ifglxtrinsertinside\else##2\fi
6780 }%
6781 \renewcommand*{\Glsxtrfullplformat}[2]{%
6782 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6783 \ifglxtrinsertinside\else##2\fi
6784 }%

```

In-line format:

```

6785 \renewcommand*{\glxtrinlinefullformat}[2]{%
6786 \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6787 \ifglxtrinsertinside\else##2\fi
6788 \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
6789 }%
6790 \renewcommand*{\glxtrinlinefullplformat}[2]{%
6791 \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6792 \ifglxtrinsertinside\else##2\fi
6793 \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
6794 }%
6795 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
6796 \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6797 \ifglxtrinsertinside\else##2\fi
6798 \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
6799 }%
6800 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
6801 \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6802 \ifglxtrinsertinside\else##2\fi
6803 \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
6804 }%
6805 }

```

**short-user-desc** Like long-postshort-user but the user supplies the description.

```

6806 \newabbreviationstyle{long-postshort-user-desc}%
6807 {%
6808 \renewcommand*{\CustomAbbreviationFields}{%
6809 name={\protect\glsfirstlongfont{\the\glslongtok}%
6810 \protect\glxtruserparen
6811 {\protect\glsabbrvfont{\the\glsshorttok}}{\the\glslabeltok}},
6812 sort={\the\glslongtok},
6813 first={\protect\glsfirstlongfont{\the\glslongtok}},%
6814 firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%
6815 plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
6816 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6817 \csdef{glxtrpostlink\glscategorylabel}{%
6818 \glxtrifwasfirstuse
6819 {%

```

```

6820      \glxtruserparen
6821      {\glsfirstabbrvuserfont{\gl Sentryshort{\glslabel}}}%
6822      {\glslabel}%
6823    }%
6824    {}%
6825  }%
6826  \glshasattribute{\the\glslabeltok}{regular}%
6827  {%
6828    \glissetattribute{\the\glslabeltok}{regular}{false}%
6829  }%
6830  {}%
6831 }%
6832 }%
6833 {%
6834   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
6835 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

6836 \newabbreviationstyle{short-postlong-user}%
6837 {%
6838   \renewcommand*{\CustomAbbreviationFields}{%
6839     name={\protect\glabbrvfont{\the\glsshorttok}},
6840     sort={\the\glsshorttok},
6841     first={\protect\glfirstlongfont{\the\glslongtok}},%
6842     firstplural={\protect\glfirstlongfont{\the\glslongpltok}},%
6843     plural={\protect\glabbrvfont{\the\glsshortpltok}},%
6844     description={\protect\glslonguserfont{\the\glslongtok}}}%
6845   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6846     \csdef{glxtrpostlink\glscategorylabel}{%
6847       \glxtrifwasfirstuse
6848       {%
6849         \glxtruserparen
6850         {\glsfirstabbrvuserfont{\gl Sentrylong{\glslabel}}}%
6851         {\glslabel}%
6852       }%
6853       {}%
6854     }%
6855     \glshasattribute{\the\glslabeltok}{regular}%
6856     {%
6857       \glissetattribute{\the\glslabeltok}{regular}{false}%
6858     }%
6859     {}%
6860   }%
6861 }%
6862 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6863 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
6864 \renewcommand*{\glabbrvfont}[1]{\glabbrvuserfont{##1}}%

```



```

6865 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
6866 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
6867 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:

```

6868 \renewcommand*{\glsxtrfullformat}[2]{%
6869   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
6870   \ifglsxtrininsertinside\else##2\fi
6871 }%
6872 \renewcommand*{\glsxtrfullplformat}[2]{%
6873   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
6874   \ifglsxtrininsertinside\else##2\fi
6875 }%
6876 \renewcommand*{\Glsxtrfullformat}[2]{%
6877   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
6878   \ifglsxtrininsertinside\else##2\fi
6879 }%
6880 \renewcommand*{\Glsxtrfullplformat}[2]{%
6881   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
6882   \ifglsxtrininsertinside\else##2\fi
6883 }%

```

In-line format:

```

6884 \renewcommand*{\glsxtrinlinefullformat}[2]{%
6885   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
6886   \ifglsxtrininsertinside\else##2\fi
6887   \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
6888 }%
6889 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
6890   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
6891   \ifglsxtrininsertinside\else##2\fi
6892   \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
6893 }%
6894 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
6895   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
6896   \ifglsxtrininsertinside\else##2\fi
6897   \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
6898 }%
6899 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
6900   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
6901   \ifglsxtrininsertinside\else##2\fi
6902   \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
6903 }%
6904 }

```

long-user-desc Like short-postlong-user but leaves the user to specify the description.

```

6905 \newabbreviationstyle{short-postlong-user-desc}%
6906 {%
6907   \renewcommand*{\CustomAbbreviationFields}{%
6908     name={\protect\glsabbrvfont{\the\glsshorttok}}%

```

```

6909         \protect\glxtruserparen
6910         {\protect\glsfirstlongfont{\the\glslongpltok}}}%
6911         {\the\glslabeltok}},
6912     sort={\the\glsshorttok},
6913     first={\protect\glsfirstlongfont{\the\glslongtok}}},%
6914     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}}},%

6915     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
6916 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6917     \csdef{glxtrpostlink\glscategorylabel}{%
6918         \glxtrifwasfirstuse
6919         {%
6920             \glxtruserparen
6921             {\glsfirstabbrvuserfont{\glsentrylong{\glslabel}}}%
6922             {\glslabel}%
6923         }%
6924     }%
6925 }%
6926 \glsattribute{\the\glslabeltok}{regular}%
6927 {%
6928     \glssetattribute{\the\glslabeltok}{regular}{false}%
6929 }%
6930 {}%
6931 }%
6932 }%
6933 {%
6934     \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
6935 }

```

short-user-desc

```

6936 \newabbreviationstyle{long-short-user-desc}%
6937 {%
6938     \GlsXtrUseAbbrStyleSetup{long-short-desc}%
6939 }%
6940 {%
6941     \GlsXtrUseAbbrStyleFmts{long-short-user}%
6942 }

```

short-long-user

```

6943 \newabbreviationstyle{short-long-user}%
6944 {%
    \glslonguserfont is used in the description since \glsdesc doesn't set the style.
6945 \renewcommand*{\CustomAbbreviationFields}{%
6946     name={\protect\glsabbrvfont{\the\glsshorttok}},
6947     sort={\the\glsshorttok},
6948     description={\protect\glslonguserfont{\the\glslongtok}}},%
6949     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
6950     \protect\glxtruserparen{\protect\glslonguserfont{\the\glslongtok}}{\the\glslabeltok}}},%
6951     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%

```

```

6952      \protect\glxtruserparen{\protect\glslonguserfont{\the\glslongpltok}}{\the\glslabeltok}},%
6953      plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

6954      \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6955      \glsattribute{\the\glslabeltok}{regular}%
6956      {%
6957      \glssetattribute{\the\glslabeltok}{regular}{false}%
6958      }%
6959      {}%
6960      }%
6961      }%
6962      {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6963      \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
6964      \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
6965      \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
6966      \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
6967      \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

The first use full form and the inline full form are the same for this style.

```

6968      \renewcommand*{\glxtrfullformat}[2]{%
6969      \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
6970      \ifglxtrinsertinside\else##2\fi
6971      \glxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
6972      }%
6973      \renewcommand*{\glxtrfullplformat}[2]{%
6974      \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6975      \ifglxtrinsertinside\else##2\fi
6976      \glxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
6977      }%
6978      \renewcommand*{\Glsxtrfullformat}[2]{%
6979      \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
6980      \ifglxtrinsertinside\else##2\fi
6981      \glxtruserparen{\glsfirstlongfont{\Glsaccesslong{##1}}}{##1}%
6982      }%
6983      \renewcommand*{\Glsxtrfullplformat}[2]{%
6984      \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6985      \ifglxtrinsertinside\else##2\fi
6986      \glxtruserparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}{##1}%
6987      }%
6988      }

```

-long-user-desc

```

6989      \newabbreviationstyle{short-long-user-desc}%
6990      {%
6991      \GlsXtrUseAbbrStyleSetup{short-long-desc}%
6992      }%
6993      {%

```

```
6994 \GlsXtrUseAbbrStyleFmts{short-long-user}%
6995 }
```

## 1.7 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:

```
6996 \let\@glxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```
6997 \renewcommand*{\markright}[1]{%
6998 \glxtrmarkhook
6999 \@glxtr@org@markright{\@glxtrinmark#1\@glxtrnotinmark}%
7000 \glxtrrestoremarkhook
7001 }
```

`\markboth` Save original definition:

```
7002 \let\@glxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
7003 \renewcommand*{\markboth}[2]{%
7004 \glxtrmarkhook
7005 \@glxtr@org@markboth
7006 {\@glxtrinmark#1\@glxtrnotinmark}%
7007 {\@glxtrinmark#2\@glxtrnotinmark}%

```

```

7008 \glsxtrrestoremarkhook
7009 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```

7010 \newcommand*{\glsxtrRevertMarks}{%
7011   \let\markright\@glsxtr@org@markright
7012   \let\markboth\@glsxtr@org@markboth
7013 }

```

\glsxtrifinmark

```

7014 \newcommand*{\glsxtrifinmark}[2]{#2}

```

\@glsxtrinmark

```

7015 \newrobustcmd*{\@glsxtrinmark}{%
7016   \let\glsxtrifinmark\@firstoftwo
7017 }

```

glsxtrnotinmark

```

7018 \newrobustcmd*{\@glsxtrnotinmark}{%
7019   \let\glsxtrifinmark\@secondoftwo
7020 }

```

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```

7021 \newcommand*{\glsxtrmarkhook}{%

```

Save current definitions:

```

7022   \let\@glsxtr@org@MakeUppercase\MakeUppercase
7023   \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
7024   \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
7025   \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
7026   \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
7027   \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
7028   \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
7029   \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
7030   \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
7031   \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
7032   \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
7033   \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
7034   \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
7035   \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
7036   \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
7037   \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
7038   \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
7039   \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
7040   \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
7041   \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
7042   \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl

```

## New definitions

```

7043 \let\glxtrifinmark\@firstoftwo
7044 \let\MakeUppercase\MakeTextUppercase
7045 \let\glxtrtitleshort\glxtrheadshort
7046 \let\glxtrtitleshortpl\glxtrheadshortpl
7047 \let\Glsxtrtitleshort\Glsxtrheadshort
7048 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
7049 \let\glxtrtitletext\glxtrheadtext
7050 \let\Glsxtrtitletext\Glsxtrheadtext
7051 \let\glxtrtitleplural\glxtrheadplural
7052 \let\Glsxtrtitleplural\Glsxtrheadplural
7053 \let\glxtrtitlefirst\glxtrheadfirst
7054 \let\Glsxtrtitlefirst\Glsxtrheadfirst
7055 \let\glxtrtitlefirstplural\glxtrheadfirstplural
7056 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
7057 \let\glxtrtitlelong\glxtrheadlong
7058 \let\glxtrtitlelongpl\glxtrheadlongpl
7059 \let\Glsxtrtitlelong\Glsxtrheadlong
7060 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
7061 \let\glxtrtitlefull\glxtrheadfull
7062 \let\glxtrtitlefullpl\glxtrheadfullpl
7063 \let\Glsxtrtitlefull\Glsxtrheadfull
7064 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
7065 }

```

`\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.)

```

7066 \newcommand*{\glxtrrestoremarkhook}{%
7067 \let\glxtrifinmark\@secondoftwo
7068 \let\MakeUppercase\@glxtr@org@MakeUppercase
7069 \let\glxtrtitleshort\@glxtr@org@glxtrtitleshort
7070 \let\glxtrtitleshortpl\@glxtr@org@glxtrtitleshortpl
7071 \let\Glsxtrtitleshort\@glxtr@org@Glsxtrtitleshort
7072 \let\Glsxtrtitleshortpl\@glxtr@org@Glsxtrtitleshortpl
7073 \let\glxtrtitletext\@glxtr@org@glxtrtitletext
7074 \let\Glsxtrtitletext\@glxtr@org@Glsxtrtitletext
7075 \let\glxtrtitleplural\@glxtr@org@glxtrtitleplural
7076 \let\Glsxtrtitleplural\@glxtr@org@Glsxtrtitleplural
7077 \let\glxtrtitlefirst\@glxtr@org@glxtrtitlefirst
7078 \let\Glsxtrtitlefirst\@glxtr@org@Glsxtrtitlefirst
7079 \let\glxtrtitlefirstplural\@glxtr@org@glxtrtitlefirstplural
7080 \let\Glsxtrtitlefirstplural\@glxtr@org@Glsxtrtitlefirstplural
7081 \let\glxtrtitlelong\@glxtr@org@glxtrtitlelong
7082 \let\glxtrtitlelongpl\@glxtr@org@glxtrtitlelongpl
7083 \let\Glsxtrtitlelong\@glxtr@org@Glsxtrtitlelong
7084 \let\Glsxtrtitlelongpl\@glxtr@org@Glsxtrtitlelongpl
7085 \let\glxtrtitlefull\@glxtr@org@glxtrtitlefull

```

```

7086 \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
7087 \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
7088 \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
7089 }

```

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.

```

7090 \newcommand*{\glsxtrheadshort}[1]{%
7091 \protect\NoCaseChange
7092 {%
7093 \glsifattribute{#1}{headuc}{true}%
7094 {%
7095 \GLSxtrshort[noindex,hyper=false]{#1}[]%
7096 }%
7097 {%
7098 \glsxtrshort[noindex,hyper=false]{#1}[]%
7099 }%
7100 }%
7101 }

```

**lsxtrtitleshort** Command to display short form of abbreviation in section title and table of contents.

```

7102 \newrobustcmd*{\lsxtrtitleshort}[1]{%
7103 \glsxtrshort[noindex,hyper=false]{#1}[]%
7104 }

```

**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.

```

7105 \newcommand*{\glsxtrheadshortpl}[1]{%
7106 \protect\NoCaseChange
7107 {%
7108 \glsifattribute{#1}{headuc}{true}%
7109 {%
7110 \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
7111 }%
7112 {%
7113 \glsxtrshortpl[noindex,hyper=false]{#1}[]%
7114 }%
7115 }%
7116 }

```

**lsxtrtitleshortpl** Command to display plural short form of abbreviation in section title and table of contents.

```

7117 \newrobustcmd*{\lsxtrtitleshortpl}[1]{%
7118 \glsxtrshortpl[noindex,hyper=false]{#1}[]%
7119 }

```

**Glsxtrheadshort** Command used to display short form in the page header with the first letter converted to upper case.

```

7120 \newcommand*{\Glsxtrheadshort}[1]{%
7121   \protect\NoCaseChange
7122   {%
7123     \glsifattribute{#1}{headuc}{true}%
7124     {%
7125       \Glsxtrshort[noindex,hyper=false]{#1}[]%
7126     }%
7127   }%
7128   \Glsxtrshort[noindex,hyper=false]{#1}[]%
7129 }%
7130 }%
7131 }

```

**lstrtitleshort** Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

7132 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
7133   \Glsxtrshort[noindex,hyper=false]{#1}[]%
7134 }

```

**sxtrheadshortpl** Command used to display plural short form in the page header with the first letter converted to upper case.

```

7135 \newcommand*{\Glsxtrheadshortpl}[1]{%
7136   \protect\NoCaseChange
7137   {%
7138     \glsifattribute{#1}{headuc}{true}%
7139     {%
7140       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
7141     }%
7142   }%
7143   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
7144 }%
7145 }%
7146 }

```

**xtrtitleshortpl** Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

7147 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
7148   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
7149 }

```

**\glsxtrheadtext** As above but for the text value.

```

7150 \newcommand*{\glsxtrheadtext}[1]{%
7151   \protect\NoCaseChange
7152   {%
7153     \glsifattribute{#1}{headuc}{true}%
7154     {%

```



```

7155     \GLStext[noindex,hyper=false]{#1}[]%
7156 }%
7157 {%
7158     \glstext[noindex,hyper=false]{#1}[]%
7159 }%
7160 }%
7161 }

```

**glsxtrtitletext** Command to display text value in section title and table of contents.

```

7162 \newrobustcmd*{\glsxtrtitletext}[1]{%
7163   \glstext[noindex,hyper=false]{#1}[]%
7164 }

```

**\Glsxtrheadtext** First letter converted to upper case

```

7165 \newcommand*{\Glsxtrheadtext}[1]{%
7166   \protect\NoCaseChange
7167   {%
7168     \glsifattribute{#1}{headuc}{true}%
7169     {%
7170       \GLStext[noindex,hyper=false]{#1}[]%
7171     }%
7172     {%
7173       \glstext[noindex,hyper=false]{#1}[]%
7174     }%
7175   }%
7176 }

```

**Glsxtrtitletext** Command to display text value in section title and table of contents with the first letter changed to upper case.

```

7177 \newrobustcmd*{\Glsxtrtitletext}[1]{%
7178   \GLstext[noindex,hyper=false]{#1}[]%
7179 }

```

**lsxtrheadplural** As above but for the plural value.

```

7180 \newcommand*{\lsxtrheadplural}[1]{%
7181   \protect\NoCaseChange
7182   {%
7183     \glsifattribute{#1}{headuc}{true}%
7184     {%
7185       \GLSplural[noindex,hyper=false]{#1}[]%
7186     }%
7187     {%
7188       \glsplural[noindex,hyper=false]{#1}[]%
7189     }%
7190   }%
7191 }

```

**sxtrtitleplural** Command to display plural value in section title and table of contents.

```

7192 \newrobustcmd*{\glxstrtitleplural}[1]{%
7193   \glsplural[noindex,hyper=false]{#1}[]%
7194 }

```

**lsxtrheadplural** Convert first letter to upper case.

```

7195 \newcommand*{\Glsxtrheadplural}[1]{%
7196   \protect\NoCaseChange
7197   {%
7198     \glsifattribute{#1}{headuc}{true}%
7199     {%
7200       \GLSplural[noindex,hyper=false]{#1}[]%
7201     }%
7202     {%
7203       \Glsplural[noindex,hyper=false]{#1}[]%
7204     }%
7205   }%
7206 }

```

**sxtrtitleplural** Command to display plural value in section title and table of contents with the first letter changed to upper case.

```

7207 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
7208   \Glsplural[noindex,hyper=false]{#1}[]%
7209 }

```

**glxstrheadfirst** As above but for the first value.

```

7210 \newcommand*{\glxstrheadfirst}[1]{%
7211   \protect\NoCaseChange
7212   {%
7213     \glsifattribute{#1}{headuc}{true}%
7214     {%
7215       \GLSfirst[noindex,hyper=false]{#1}[]%
7216     }%
7217     {%
7218       \glsfirst[noindex,hyper=false]{#1}[]%
7219     }%
7220   }%
7221 }

```

**lsxtrtitlefirst** Command to display first value in section title and table of contents.

```

7222 \newrobustcmd*{\glxtrtitlefirst}[1]{%
7223   \glsfirst[noindex,hyper=false]{#1}[]%
7224 }

```

**Glsxtrheadfirst** First letter converted to upper case

```

7225 \newcommand*{\Glsxtrheadfirst}[1]{%
7226   \protect\NoCaseChange
7227   {%
7228     \glsifattribute{#1}{headuc}{true}%

```

```

7229  {%
7230    \GLSfirst[noindex,hyper=false]{#1}[]%
7231  }%
7232  {%
7233    \GLSfirst[noindex,hyper=false]{#1}[]%
7234  }%
7235 }%
7236 }

```

**lsxtrtitlefirst** Command to display first value in section title and table of contents with the first letter changed to upper case.

```

7237 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
7238   \GLSfirst[noindex,hyper=false]{#1}[]%
7239 }

```

**headfirstplural** As above but for the firstplural value.

```

7240 \newcommand*{\Glsxtrheadfirstplural}[1]{%
7241   \protect\NoCaseChange
7242   {%
7243     \glsifattribute{#1}{headuc}{true}%
7244     {%
7245       \GLSfirstplural[noindex,hyper=false]{#1}[]%
7246     }%
7247     {%
7248       \glsfirstplural[noindex,hyper=false]{#1}[]%
7249     }%
7250   }%
7251 }

```

**itlefirstplural** Command to display firstplural value in section title and table of contents.

```

7252 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
7253   \glsfirstplural[noindex,hyper=false]{#1}[]%
7254 }

```

**headfirstplural** First letter converted to upper case

```

7255 \newcommand*{\Glsxtrheadfirstplural}[1]{%
7256   \protect\NoCaseChange
7257   {%
7258     \glsifattribute{#1}{headuc}{true}%
7259     {%
7260       \GLSfirstplural[noindex,hyper=false]{#1}[]%
7261     }%
7262     {%
7263       \Glsfirstplural[noindex,hyper=false]{#1}[]%
7264     }%
7265   }%
7266 }

```

`\titlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```
7267 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
7268   \Glsfirstplural[noindex,hyper=false]{#1}[]%
7269 }
```

`\glxtrheadlong` Command used to display long form in the page header.

```
7270 \newcommand*{\glxtrheadlong}[1]{%
7271   \protect\NoCaseChange
7272   {%
7273     \glsifattribute{#1}{headuc}{true}%
7274     {%
7275       \GLSxtrlong[noindex,hyper=false]{#1}[]%
7276     }%
7277     {%
7278       \glxtrlong[noindex,hyper=false]{#1}[]%
7279     }%
7280   }%
7281 }
```

`\glxtrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```
7282 \newrobustcmd*{\glxtrtitlelong}[1]{%
7283   \glxtrlong[noindex,hyper=false]{#1}[]%
7284 }
```

`\glxtrheadlongpl` 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.

```
7285 \newcommand*{\glxtrheadlongpl}[1]{%
7286   \protect\NoCaseChange
7287   {%
7288     \glsifattribute{#1}{headuc}{true}%
7289     {%
7290       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
7291     }%
7292     {%
7293       \glxtrlongpl[noindex,hyper=false]{#1}[]%
7294     }%
7295   }%
7296 }
```

`\glxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```
7297 \newrobustcmd*{\glxtrtitlelongpl}[1]{%
7298   \glxtrlongpl[noindex,hyper=false]{#1}[]%
7299 }
```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```

7300 \newcommand*{\Glsxtrheadlong}[1]{%
7301   \protect\NoCaseChange
7302   {%
7303     \glsifattribute{#1}{headuc}{true}%
7304     {%
7305       \GLSxtrlong[noindex,hyper=false]{#1}[]%
7306     }%
7307     {%
7308       \GLSxtrlong[noindex,hyper=false]{#1}[]%
7309     }%
7310   }%
7311 }

```

**Glsxtrtitlelong** Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

7312 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
7313   \GLSxtrlong[noindex,hyper=false]{#1}[]%
7314 }

```

**lsxtrheadlongpl** Command used to display plural long form in the page header with the first letter converted to upper case.

```

7315 \newcommand*{\Glsxtrheadlongpl}[1]{%
7316   \protect\NoCaseChange
7317   {%
7318     \glsifattribute{#1}{headuc}{true}%
7319     {%
7320       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
7321     }%
7322     {%
7323       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
7324     }%
7325   }%
7326 }

```

**sxtrtitlelongpl** Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

7327 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
7328   \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
7329 }

```

**\glsxtrheadfull** Command used to display full form in the page header.

```

7330 \newcommand*{\glsxtrheadfull}[1]{%
7331   \protect\NoCaseChange
7332   {%
7333     \glsifattribute{#1}{headuc}{true}%
7334     {%
7335       \GLSxtrfull[noindex,hyper=false]{#1}[]%
7336     }%

```

```

7337   {%
7338     \glsxtrfull[noindex,hyper=false]{#1}[]%
7339   }%
7340 }%
7341 }

```

**glsxtrtitlefull** Command to display full form of abbreviation in section title and table of contents.

```

7342 \newrobustcmd*{\glsxtrtitlefull}[1]{%
7343   \glsxtrfull[noindex,hyper=false]{#1}[]%
7344 }

```

**lsxtrheadfullpl** Command used to display plural full form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrfullpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```

7345 \newcommand*{\lsxtrheadfullpl}[1]{%
7346   \protect\NoCaseChange
7347   {%
7348     \glsifattribute{#1}{headuc}{true}%
7349     {%
7350       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
7351     }%
7352     {%
7353       \glsxtrfullpl[noindex,hyper=false]{#1}[]%
7354     }%
7355   }%
7356 }

```

**sxtrtitlefullpl** Command to display plural full form of abbreviation in section title and table of contents.

```

7357 \newrobustcmd*{\lsxtrtitlefullpl}[1]{%
7358   \glsxtrfullpl[noindex,hyper=false]{#1}[]%
7359 }

```

**\GLsxtrheadfull** Command used to display full form in the page header with the first letter converted to upper case.

```

7360 \newcommand*{\GLsxtrheadfull}[1]{%
7361   \protect\NoCaseChange
7362   {%
7363     \glsifattribute{#1}{headuc}{true}%
7364     {%
7365       \GLSxtrfull[noindex,hyper=false]{#1}[]%
7366     }%
7367     {%
7368       \GLsxtrfull[noindex,hyper=false]{#1}[]%
7369     }%
7370   }%
7371 }

```

**GLsxtrtitlefull** Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

7372 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
7373   \Glsxtrfull[noindex,hyper=false]{#1}[]%
7374 }

```

`\lsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```

7375 \newcommand*{\Glsxtrheadfullpl}[1]{%
7376   \protect\NoCaseChange
7377   {%
7378     \glsifattribute{#1}{headuc}{true}%
7379     {%
7380       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
7381     }%
7382     {%
7383       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
7384     }%
7385   }%
7386 }

```

`\sxttrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

7387 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
7388   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
7389 }

```

`\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.

```

7390 \ifdef\texorpdfstring
7391 {
7392   \newcommand*{\glsfmtshort}[1]{%
7393     \texorpdfstring
7394       {\glsxtrtitleshort{#1}}%
7395       {\glsentryshort{#1}}%
7396   }
7397 }
7398 {
7399   \newcommand*{\glsfmtshort}[1]{%
7400     \glsxtrtitleshort{#1}}
7401 }

```

Similarly for the plural version.

`\glsfmtshortpl`

```

7402 \ifdef\texorpdfstring
7403 {
7404   \newcommand*{\glsfmtshortpl}[1]{%
7405     \texorpdfstring
7406       {\glsxtrtitleshortpl{#1}}%
7407       {\glsentryshortpl{#1}}%

```

```

7408 }
7409 }
7410 {
7411   \newcommand*{\glsfmtshortpl}[1]{%
7412     \glsxtrtitleshortpl{#1}}
7413 }

```

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).

```

7414 \ifdef\texorpdfstring
7415 {
7416   \newcommand*{\Glsfmtshort}[1]{%
7417     \texorpdfstring
7418       {\Glsxtrtitleshort{#1}}%
7419     {\glsentryshort{#1}}%
7420   }
7421 }
7422 {
7423   \newcommand*{\Glsfmtshort}[1]{%
7424     \Glsxtrtitleshort{#1}}
7425 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

7426 \ifdef\texorpdfstring
7427 {
7428   \newcommand*{\Glsfmtshortpl}[1]{%
7429     \texorpdfstring
7430       {\Glsxtrtitleshortpl{#1}}%
7431     {\glsentryshortpl{#1}}%
7432   }
7433 }
7434 {
7435   \newcommand*{\Glsfmtshortpl}[1]{%
7436     \Glsxtrtitleshortpl{#1}}
7437 }

```

`\glsfmttext` As above but for the text value.

```

7438 \ifdef\texorpdfstring
7439 {
7440   \newcommand*{\glsfmttext}[1]{%
7441     \texorpdfstring
7442       {\glsxtrtitletext{#1}}%
7443     {\glsentrytext{#1}}%
7444   }
7445 }
7446 {
7447   \newcommand*{\glsfmttext}[1]{%

```



```

7448 \glstrtrtitletext{#1}}
7449 }

```

`\Glsfmttext` First letter converted to upper case.

```

7450 \ifdef\teorpdfstring
7451 {
7452   \newcommand*{\Glsfmttext}[1]{%
7453     \teorpdfstring
7454     {\Glsxtrtitletext{#1}}}%
7455     {\glstrytext{#1}}}%
7456   }
7457 }
7458 {
7459   \newcommand*{\Glsfmttext}[1]{%
7460     \Glsxtrtitletext{#1}}
7461 }

```

`\glsfmtplural` As above but for the plural value.

```

7462 \ifdef\teorpdfstring
7463 {
7464   \newcommand*{\glsfmtplural}[1]{%
7465     \teorpdfstring
7466     {\Glsxtrtitleplural{#1}}}%
7467     {\glstryplural{#1}}}%
7468   }
7469 }
7470 {
7471   \newcommand*{\glsfmtplural}[1]{%
7472     \Glsxtrtitleplural{#1}}
7473 }

```

`\Glsfmtplural` First letter converted to upper case.

```

7474 \ifdef\teorpdfstring
7475 {
7476   \newcommand*{\Glsfmtplural}[1]{%
7477     \teorpdfstring
7478     {\Glsxtrtitleplural{#1}}}%
7479     {\glstryplural{#1}}}%
7480   }
7481 }
7482 {
7483   \newcommand*{\Glsfmtplural}[1]{%
7484     \Glsxtrtitleplural{#1}}
7485 }

```

`\glsfmtfirst` As above but for the first value.

```

7486 \ifdef\teorpdfstring
7487 {
7488   \newcommand*{\glsfmtfirst}[1]{%

```

```

7489 \texorpdfstring
7490 {\glxtrtitlefirst{#1}}}%
7491 {\glsentryfirst{#1}}}%
7492 }
7493 }
7494 {
7495 \newcommand*{\Glsfmtfirst}[1]{%
7496 \glxtrtitlefirst{#1}}
7497 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

7498 \ifdef\texorpdfstring
7499 {
7500 \newcommand*{\Glsfmtfirst}[1]{%
7501 \texorpdfstring
7502 {\Glsxtrtitlefirst{#1}}}%
7503 {\glsentryfirst{#1}}}%
7504 }
7505 }
7506 {
7507 \newcommand*{\Glsfmtfirst}[1]{%
7508 \Glsxtrtitlefirst{#1}}
7509 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

7510 \ifdef\texorpdfstring
7511 {
7512 \newcommand*{\glsfmtfirstpl}[1]{%
7513 \texorpdfstring
7514 {\glxtrtitlefirstplural{#1}}}%
7515 {\glsentryfirstplural{#1}}}%
7516 }
7517 }
7518 {
7519 \newcommand*{\glsfmtfirstpl}[1]{%
7520 \glxtrtitlefirstplural{#1}}
7521 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

7522 \ifdef\texorpdfstring
7523 {
7524 \newcommand*{\Glsfmtfirstpl}[1]{%
7525 \texorpdfstring
7526 {\Glsxtrtitlefirstplural{#1}}}%
7527 {\glsentryfirstplural{#1}}}%
7528 }
7529 }
7530 {
7531 \newcommand*{\Glsfmtfirstpl}[1]{%

```

```

7532 \Glsxtrtitlefirstplural{#1}}
7533 }

```

`\glsfmtlong` As above but for the long value.

```

7534 \ifdef\txorpdfstring
7535 {
7536   \newcommand*{\glsfmtlong}[1]{%
7537     \txorpdfstring
7538     {\glsxtrtitlelong{#1}}%
7539     {\glsentrylong{#1}}%
7540   }
7541 }
7542 {
7543   \newcommand*{\glsfmtlong}[1]{%
7544     \glsxtrtitlelong{#1}}
7545 }

```

`\Glsfmtlong` First letter converted to upper case.

```

7546 \ifdef\txorpdfstring
7547 {
7548   \newcommand*{\Glsfmtlong}[1]{%
7549     \txorpdfstring
7550     {\Glsxtrtitlelong{#1}}%
7551     {\glsentrylong{#1}}%
7552   }
7553 }
7554 {
7555   \newcommand*{\Glsfmtlong}[1]{%
7556     \Glsxtrtitlelong{#1}}
7557 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

7558 \ifdef\txorpdfstring
7559 {
7560   \newcommand*{\glsfmtlongpl}[1]{%
7561     \txorpdfstring
7562     {\glsxtrtitlelongpl{#1}}%
7563     {\glsentrylongpl{#1}}%
7564   }
7565 }
7566 {
7567   \newcommand*{\glsfmtlongpl}[1]{%
7568     \glsxtrtitlelongpl{#1}}
7569 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

7570 \ifdef\txorpdfstring
7571 {
7572   \newcommand*{\Glsfmtlongpl}[1]{%

```

```

7573 \texorpdfstring
7574 {\Glsxtrtitlelongpl{#1}}%
7575 {\glsentrylongpl{#1}}%
7576 }
7577 }
7578 {
7579 \newcommand*{\Glsfmtlongpl}[1]{%
7580 \Glsxtrtitlelongpl{#1}}
7581 }

```

`\glsfmtfull` In-line full format.

```

7582 \ifdef\texorpdfstring
7583 {
7584 \newcommand*{\glsfmtfull}[1]{%
7585 \texorpdfstring
7586 {\glsxtrtitlefull{#1}}%
7587 {\glsxtrinlinefullformat{#1}{}}%
7588 }
7589 }
7590 {
7591 \newcommand*{\glsfmtfull}[1]{%
7592 \glsxtrtitlefull{#1}}
7593 }

```

`\Glsfmtfull` First letter converted to upper case.

```

7594 \ifdef\texorpdfstring
7595 {
7596 \newcommand*{\Glsfmtfull}[1]{%
7597 \texorpdfstring
7598 {\Glsxtrtitlefull{#1}}%
7599 {\Glsxtrinlinefullformat{#1}{}}%
7600 }
7601 }
7602 {
7603 \newcommand*{\Glsfmtfull}[1]{%
7604 \Glsxtrtitlefull{#1}}
7605 }

```

`\glsfmtfullpl` In-line full plural format.

```

7606 \ifdef\texorpdfstring
7607 {
7608 \newcommand*{\glsfmtfullpl}[1]{%
7609 \texorpdfstring
7610 {\glsxtrtitlefullpl{#1}}%
7611 {\glsxtrinlinefullplformat{#1}{}}%
7612 }
7613 }
7614 {
7615 \newcommand*{\glsfmtfullpl}[1]{%

```

```

7616 \glsxtrtitlefullpl{#1}}
7617 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

7618 \ifdef\teorpdfstring
7619 {
7620 \newcommand*{\Glsfmtfullpl}[1]{%
7621 \teorpdfstring
7622 {\Glsxtrtitlefullpl{#1}}%
7623 {\Glsxtrinlinefullplformat{#1}{}}%
7624 }
7625 }
7626 {
7627 \newcommand*{\Glsfmtfullpl}[1]{%
7628 \Glsxtrtitlefullpl{#1}}
7629 }

```

## 1.8 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

7630 \newcommand*{\RequireGlossariesExtraLang}[1]{%
7631 \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
7632 }

```

`sariesExtraLang`

```

7633 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
7634 \ProvidesFile{glossariesxtr-#1.ldf}%
7635 }

```

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.)

```

7636 \@ifpackageloaded{tracklang}
7637 {%
7638 \AnyTrackedLanguages
7639 {%
7640 \ForEachTrackedDialect{\this@dialect}{%
7641 \IfTrackedLanguageFileExists{\this@dialect}%
7642 {glossariesxtr-}% prefix
7643 {.ldf}%
7644 {%
7645 \RequireGlossariesExtraLang{\CurrentTrackedTag}%
7646 }%
7647 }%

```

```

7648      }%
7649      }%
7650      }%
7651      {}%
7652  }
7653 {}

```

Load glossaries-extra-stylemods if required.

```
7654 \@glsxtr@redefstyles
```

and set the style:

```
7655 \@glsxtr@do@style
```

## 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:

```
7656 \NeedsTeXFormat{LaTeX2e}
7657 \ProvidesPackage{glossaries-extra-stylemods}[2017/06/15 v1.16 (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`

```
7658 \newcommand*{\@glxtr@loadstyles}{%
7659 \DeclareOption*{%
7660   \IfFileExists{glossary-\CurrentOption.sty}%
7661   {\eappto\@glxtr@loadstyles{%
7662     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
7663   {\PackageError{glossaries-extra-styles}%
7664     {Unknown option '\CurrentOption'}{}}
7665 }
```

Process the package options:

```
7666 \ProcessOptions
```

Load the required packages:

```
7667 \@glxtr@loadstyles
```

Adjust the styles that the post description hook added, but only for styles that have already been defined. All the tree styles in `glossary-tree` include the post description hook, so they don't require adjustment. Similarly for `glossary-mcols` which builds on the tree styles.

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

`ewglossarystyle`

```
7668 \providecommand{\renewglossarystyle}[2]{%
7669   \ifcsundef{@glstyle@#1}%
7670   {%
7671     \PackageError{glossaries}{Glossary style '#1' isn't already defined}{}%

```

```

7672 }%
7673 {%
7674   \csdef{@glsstyle@#1}{#2}%
7675 }%
7676 }

```

## 2.2 List-Like Styles

The list-like styles mostly already use the post description hook. Only the listdotted style need modifying.

```

7677 \ifdefined{@glsstyle@listdotted}
7678 {%
7679   \renewglossarystyle{listdotted}{%
7680     \setglossarystyle{list}%
7681     \renewcommand*{\glossentry}[2]{%
7682       \item[]\makebox[\glslistdottedwidth][l]{%
7683         \glsentryitem{##1}%
7684         \glstarget{##1}{\glossentryname{##1}}%
7685         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
7686         \glossentrydesc{##1}\glspostdescription}%
7687     \renewcommand*{\subglossentry}[3]{%
7688       \item[]\makebox[\glslistdottedwidth][l]{%
7689         \glssubentryitem{##2}%
7690         \glstarget{##2}{\glossentryname{##2}}%
7691         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
7692         \glossentrydesc{##2}\glspostdescription}%
7693   }
7694 }
7695 {}

```

The sublistdotted style doesn't display the description for top-level entries. Sub-level entries use the listdottedstyle.

## 2.3 Longtable Styles

The three and four column styles require adjustment, but not the two column styles.

```

7696 \ifcsdef{@glsstyle@long3col}
7697 {%
7698   \renewglossarystyle{long3col}{%
7699     \renewenvironment{theglossary}%
7700       {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
7701       {\end{longtable}}}%
7702   \renewcommand*{\glossaryheader}{}%
7703   \renewcommand*{\glsgroupheading}[1]{}%
7704   \renewcommand{\glossentry}[2]{%
7705     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7706     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline

```



```

7707 }%
7708 \renewcommand{\subglossentry}[3]{%
7709     &
7710     \glssubentryitem{##2}%
7711     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7712     ##3\tabularnewline
7713 }%
7714 \renewcommand*\{ \glsgroupskip }{%
7715     \ifglsgnognogroupskip\else & &\tabularnewline\fi}%
7716 }
7717 }
7718 {}

```

Four column style:

```

7719 \ifcsdef{@glsstyle@long4col}
7720 {%
7721     \renewglossarystyle{long4col}{%
7722         \renewenvironment{theglossary}%
7723             {\begin{longtable}{llll}}%
7724             {\end{longtable}}%
7725         \renewcommand*\{ \glossaryheader }{%
7726             \renewcommand*\{ \glsgroupheading }[1]{%
7727                 \renewcommand{\glossentry}[2]{%
7728                     \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7729                     \glossentrydesc{##1}\glspostdescription &
7730                     \glossentrysymbol{##1} &
7731                     ##2\tabularnewline
7732                 }%
7733                 \renewcommand{\subglossentry}[3]{%
7734                     &
7735                     \glssubentryitem{##2}%
7736                     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7737                     \glossentrysymbol{##2} & ##3\tabularnewline
7738                 }%
7739                 \renewcommand*\{ \glsgroupskip }{%
7740                     \ifglsgnognogroupskip\else & &\tabularnewline\fi}%
7741             }
7742         }
7743     }

```

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, but not the two column styles.

```

7744 \ifcsdef{@glsstyle@longragged3col}
7745 {%
7746     \renewglossarystyle{longragged3col}{%

```

```

7747 \renewenvironment{theglossary}%
7748   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%
7749     >{\raggedright}p{\glspagelistwidth}}}%
7750   {\end{longtable}}}%
7751 \renewcommand*{\glossaryheader}{}%
7752 \renewcommand*{\glsgroupheading}[1]{}%
7753 \renewcommand{\glossentry}[2]{%
7754   \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7755   \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
7756 }%
7757 \renewcommand{\subglossentry}[3]{%
7758   &
7759   \glssubentryitem{##2}%
7760   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7761   ##3\tabularnewline
7762 }%
7763 \renewcommand*{\glsgroupskip}{%
7764   \ifglsgnোগroupskip\else & &\tabularnewline\fi}%
7765 }
7766 }
7767 {}

```

Four column style:

```

7768 \ifcsdef{@glstyle@altlongragged4col}
7769 {%
7770   \renewglossarystyle{altlongragged4col}{%
7771     \renewenvironment{theglossary}%
7772       {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}1%
7773         >{\raggedright}p{\glspagelistwidth}}}%
7774       {\end{longtable}}}%
7775     \renewcommand*{\glossaryheader}{}%
7776     \renewcommand*{\glsgroupheading}[1]{}%
7777     \renewcommand{\glossentry}[2]{%
7778       \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7779       \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
7780       ##2\tabularnewline
7781     }%
7782     \renewcommand{\subglossentry}[3]{%
7783       &
7784       \glssubentryitem{##2}%
7785       \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7786       \glossentrysymbol{##2} & ##3\tabularnewline
7787     }%
7788     \renewcommand*{\glsgroupskip}{%
7789       \ifglsgnোগroupskip\else & & &\tabularnewline\fi}%
7790   }
7791 }
7792 {}

```

## 2.5 Supertabular Styles

The three and four column styles require adjustment, but not the two column styles.

```
7793 \ifcsdef{@glsstyle@super3col}
7794 {%
7795   \renewglossarystyle{super3col}{%
7796     \renewenvironment{theglossary}%
7797       {\tablehead{}}\tabletail{}}%
7798     \begin{supertabular}{\lp{@glsdescwidth}p{@glspagelistwidth}}}%
7799     {\end{supertabular}}}%
7800   \renewcommand*{\glossaryheader}{}%
7801   \renewcommand*{\glsgroupheading}[1]{}%
7802   \renewcommand{\glossentry}[2]{%
7803     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7804     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
7805   }%
7806   \renewcommand{\subglossentry}[3]{%
7807     &
7808     \glssubentryitem{##2}%
7809     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7810     ##3\tabularnewline
7811   }%
7812   \renewcommand*{\glsgroupskip}{%
7813     \ifglsnogroupskip\else & &\tabularnewline\fi}%
7814 }
7815 }
7816 {}
```

Four column styles:

```
7817 \ifcsdef{@glsstyle@super4col}
7818 {%
7819   \renewglossarystyle{super4col}{%
7820     \renewenvironment{theglossary}%
7821       {\tablehead{}}\tabletail{}}%
7822     \begin{supertabular}{\llll}}}%
7823     \end{supertabular}}}%
7824   \renewcommand*{\glossaryheader}{}%
7825   \renewcommand*{\glsgroupheading}[1]{}%
7826   \renewcommand{\glossentry}[2]{%
7827     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7828     \glossentrydesc{##1}\glspostdescription &
7829     \glossentrysymbol{##1} & ##2\tabularnewline
7830   }%
7831   \renewcommand{\subglossentry}[3]{%
7832     &
7833     \glssubentryitem{##2}%
7834     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7835     \glossentrysymbol{##2} & ##3\tabularnewline
7836   }%
7837   \renewcommand*{\glsgroupskip}{%
```

```

7838     \ifglsnogroupskip\else & & \tabularnewline\fi}%
7839   }
7840 }
7841 {}

```

## 2.6 Super Ragged Styles

The three and four column styles require adjustment, but not the two column styles.

```

7842 \ifcsdef{@glsstyle@superragged3col}
7843 {%
7844   \renewglossarystyle{superragged3col}{%
7845     \renewenvironment{theglossary}%
7846       {\tablehead{}\tabletail}%
7847       \begin{supertabular}{1>\raggedright}p{\glsdescwidth}%
7848         >\raggedright}p{\glspagelistwidth}}}%
7849     {\end{supertabular}}}%
7850   \renewcommand*{\glossaryheader}{}%
7851   \renewcommand*{\glsgroupheading}[1]{}%
7852   \renewcommand{\glossentry}[2]{%
7853     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7854     \glossentrydesc{##1}\glspostdescription &
7855     ##2\tabularnewline
7856   }%
7857   \renewcommand{\subglossentry}[3]{%
7858     &
7859     \glssubentryitem{##2}%
7860     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7861     ##3\tabularnewline
7862   }%
7863   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
7864     &\tabularnewline\fi}%
7865 }
7866 }
7867 {}

```

Four columns:

```

7868 \ifcsdef{@glsstyle@altsuperragged4col}
7869 {%
7870   \renewglossarystyle{altsuperragged4col}{%
7871     \renewenvironment{theglossary}%
7872       {\tablehead{}\tabletail}%
7873       \begin{supertabular}{1>\raggedright}p{\glsdescwidth}l%
7874         >\raggedright}p{\glspagelistwidth}}}%
7875     {\end{supertabular}}}%
7876   \renewcommand*{\glossaryheader}{}%
7877   \renewcommand{\glossentry}[2]{%
7878     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7879     \glossentrydesc{##1}\glspostdescription &
7880     \glossentrysymbol{##1} & ##2\tabularnewline

```

```

7881 }%
7882 \renewcommand{\subglossentry}[3]{%
7883     &
7884     \glssubentryitem{##2}%
7885     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7886     \glossentrysymbol{##2} & ##3\tabularnewline
7887 }%
7888 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & &
7889     &\tabularnewline\fi}%
7890 }
7891 }
7892 {}

```

## 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`.

```

7893 \ifdef{\@glsstyle@inline}
7894 {%
7895     \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
7896     Just use \glxtrpostdescription instead of \glspostdescription.
7897     \renewcommand*{\glsinlinedescformat}[3]{%
7898         \space#1\glxtrpostdescription}
7899     \renewcommand*{\glsinlinesubdescformat}[3]{%
7900         #1\glxtrpostdescription}
7901 }

```

## 2.8 Tree Styles

The `alttree` style is redefined to make it easier to made minor adjustments.

```

7902 \ifdef{\@glsstyle@alttree}
7903 {%

```

Only redefine this style if it's already been defined.

mbolDescLocation

```
\glxtraltrtreeSymbolDescLocation{\label}{\location list}
```

Layout the symbol, description and location for top-level entries.

```

7904 \newcommand{\glxtraltrtreeSymbolDescLocation}[2]{%
7905     {%
7906         \let\par\glxtrAltTreePar

```

```

7907      \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
7908      \glossentrydesc{#1}\glspostdescription \space #2\par
7909  }%
7910  }

```

`\trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

7911  \newlength\glxstrAltTreeIndent

```

`\lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

7912  \newcommand{\glxstrAltTreePar}{%
7913    @@par
7914    \glxstrAltTreeSetHangIndent
7915    \setlength{\parindent}{\dimexpr\hangindent+\glxstrAltTreeIndent}%
7916  }

```

`\symbolDescLocation` `\glxstralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

7917  \newcommand{\glxstralttreeSubSymbolDescLocation}[3]{%
7918    \glxstralttreeSymbolDescLocation{#2}{#3}%
7919  }

```

`\trtreeindent` 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.

```

7920  \newlength\glxstrtreeindent

```

`\glxstralttreeInit` User-level initialisation for the alttree style.

```

7921  \newcommand*{\glxstralttreeInit}{%
7922    \settowidth{\glxstrtreeindent}{\glstreenamfmt{\glsgetwidestname\space}}%
7923    \glxstrAltTreeIndent=\parindent
7924  }

```

`\eglissetwidest` The original `\glissetwidest` only uses `\def`. This uses `\protected@csedef`.

```

7925  \newcommand*{\eglissetwidest}[2][0]{%
7926    \protected@csedef{@glswidestname\romannumeral#1}{#2}%
7927  }

```

`\xglissetwidest` Like the above but uses `\protected@csxdef`.

```

7928  \newcommand*{\xglissetwidest}[2][0]{%
7929    \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
7930  }

```

`\lsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

7931  \newcommand*{\lsgetwidestname}{\@glswidestname}

```

etwidestsubname Provide a user-level macro to obtain the widest sub-entry name.

```
7932 \newcommand*\glsgetwidestsubname}[1]{%
7933   \ifcsundef{@glswidestname\romannumeral#1}%
7934     {\@glswidestname}%
7935     {\csuse{@glswidestname\romannumeral#1}}%
7936 }
```

estTopLevelName CamelCase is easier for long command names. Provide a CamelCase synonym of \glsfindwidesttoplevelname

```
7937 \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.

```
7938 \newrobustcmd*\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
7939   \dimen@=0pt\relax
7940   \gls@tmplen=0pt\relax
7941   \foralllglossaries[#1]{\@gls@type}%
7942   {%
7943     \forglsentries[\@gls@type]{\@glo@label}%
7944     {%
7945       \ifglsused{\@glo@label}%
7946       {%
7947         \ifglshasparent{\@glo@label}%
7948         {}%
7949         {%
7950           \settowidth{\dimen@}%
7951             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
7952           \ifdim\dimen@>\gls@tmplen
7953             \gls@tmplen=\dimen@
7954             \eglssetwidest{\glsentryname{\@glo@label}}%
7955           \fi
7956         }%
7957       }%
7958     }%
7959   }%
7960 }%
7961 }
```

destUsedAnyName Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```
7962 \newrobustcmd*\glsFindWidestUsedAnyName}[1][\@glo@types]{%
7963   \dimen@=0pt\relax
7964   \gls@tmplen=0pt\relax
7965   \foralllglossaries[#1]{\@gls@type}%
7966   {%
7967     \forglsentries[\@gls@type]{\@glo@label}%
7968     {%
```

```

7969     \ifglused{\@glo@label}%
7970     {%
7971         \settowidth{\dimen@}%
7972         {\glstreenamefmt{\glentryname{\@glo@label}}}%
7973         \ifdim\dimen@>\gls@tmplen
7974             \gls@tmplen=\dimen@
7975             \eglssetwidest{\glentryname{\@glo@label}}%
7976         \fi
7977     }%
7978     {%
7979     }%
7980 }%
7981 }

```

**ndWidestAnyName** Like the above but doesn't check if the entry has been used.

```

7982 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
7983     \dimen@=0pt\relax
7984     \gls@tmplen=0pt\relax
7985     \forallglossaries[#1]{\@gls@type}%
7986     {%
7987         \forglentries[\@gls@type]{\@glo@label}%
7988         {%
7989             \settowidth{\dimen@}%
7990             {\glstreenamefmt{\glentryname{\@glo@label}}}%
7991             \ifdim\dimen@>\gls@tmplen
7992                 \gls@tmplen=\dimen@
7993                 \eglssetwidest{\glentryname{\@glo@label}}%
7994             \fi
7995         }%
7996     }%
7997 }

```

**estUsedLevelTwo** This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```

7998 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
7999     \dimen@=0pt\relax
8000     \dimen@i=0pt\relax
8001     \dimen@ii=0pt\relax
8002     \forallglossaries[#1]{\@gls@type}%
8003     {%
8004         \forglentries[\@gls@type]{\@glo@label}%
8005         {%
8006             \ifglused{\@glo@label}%
8007             {%
8008                 \ifglshasparent{\@glo@label}%
8009                 {%
8010                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
8011                     \ifglshasparent{\@glo@parent}%
8012                 }%

```



```

8013         \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}}%
8014         \ifglshasparent{\@glo@parent}%
8015         {%
8016         {%
8017             \settowidth{\gls@tmplen}%
8018             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8019             \ifdim\gls@tmplen>\dimen@ii
8020             \dimen@ii=\gls@tmplen
8021             \eglssetwidest[2]{\glsentryname{\@glo@label}}%
8022             \fi
8023         }%
8024     }%
8025     {%
8026         \settowidth{\gls@tmplen}%
8027         {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8028         \ifdim\gls@tmplen>\dimen@i
8029         \dimen@i=\gls@tmplen
8030         \eglssetwidest[1]{\glsentryname{\@glo@label}}%
8031         \fi
8032     }%
8033 }%
8034 {%
8035     \settowidth{\gls@tmplen}%
8036     {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8037     \ifdim\gls@tmplen>\dimen@
8038     \dimen@=\gls@tmplen
8039     \eglssetwidest{\glsentryname{\@glo@label}}%
8040     \fi
8041 }%
8042 }%
8043 {}%
8044 }%
8045 }%
8046 }

```

`dWidestLevelTwo` This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

8047 \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
8048     \dimen@=0pt\relax
8049     \dimen@i=0pt\relax
8050     \dimen@ii=0pt\relax
8051     \foralllglossaries[#1]{\@gls@type}%
8052     {%
8053         \forglsentries[\@gls@type]{\@glo@label}%
8054         {%
8055             \ifglshasparent{\@glo@label}%
8056             {%
8057                 \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
8058                 \ifglshasparent{\@glo@parent}%
8059                 {%

```

```

8060      \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
8061      \ifglshasparent{\@glo@parent}%
8062      {%
8063      {%
8064          \settowidth{\gls@tmplen}%
8065              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8066          \ifdim\gls@tmplen>\dimen@ii
8067              \dimen@ii=\gls@tmplen
8068              \eglssetwidest[2]{\glsentryname{\@glo@label}}%
8069          \fi
8070      }%
8071      }%
8072      {%
8073          \settowidth{\gls@tmplen}%
8074              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8075          \ifdim\gls@tmplen>\dimen@i
8076              \dimen@i=\gls@tmplen
8077              \eglssetwidest[1]{\glsentryname{\@glo@label}}%
8078          \fi
8079      }%
8080      }%
8081      {%
8082          \settowidth{\gls@tmplen}%
8083              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8084          \ifdim\gls@tmplen>\dimen@
8085              \dimen@=\gls@tmplen
8086              \eglssetwidest{\glsentryname{\@glo@label}}%
8087          \fi
8088      }%
8089      }%
8090      }%
8091      }

```

**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.

```

8092  \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
8093      \dimen@=0pt\relax
8094      \gls@tmplen=0pt\relax
8095      #2=0pt\relax
8096      \forallglossaries[#1]{\@gls@type}%
8097      {%
8098          \forglsentries[\@gls@type]{\@glo@label}%
8099          {%
8100              \ifglused{\@glo@label}%
8101              {%
8102                  \settowidth{\dimen@}%
8103                      {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8104                  \ifdim\dimen@>\gls@tmplen
8105                      \gls@tmplen=\dimen@

```

```

8106         \eglssetwidest{\glsentryname{\@glo@label}}}%
8107         \fi
8108         \settowidth{\dimen@}%
8109         {\glsentrysymbol{\@glo@label}}}%
8110         \ifdim\dimen@>#2\relax
8111             #2=\dimen@
8112         \fi
8113     }%
8114     {}%
8115 }%
8116 }%
8117 }

```

**stAnyNameSymbol** Like the above but doesn't check if the entry has been used.

```

8118 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
8119     \dimen@=0pt\relax
8120     \gls@tmplen=0pt\relax
8121     #2=0pt\relax
8122     \forallglossaries[#1]{\@gls@type}%
8123     {%
8124         \forglsentries[\@gls@type]{\@glo@label}%
8125         {%
8126             \settowidth{\dimen@}%
8127             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
8128             \ifdim\dimen@>\gls@tmplen
8129                 \gls@tmplen=\dimen@
8130                 \eglssetwidest{\glsentryname{\@glo@label}}}%
8131             \fi
8132             \settowidth{\dimen@}%
8133             {\glsentrysymbol{\@glo@label}}}%
8134             \ifdim\dimen@>#2\relax
8135                 #2=\dimen@
8136             \fi
8137         }%
8138     }%
8139 }

```

**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.

```

8140 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
8141     \dimen@=0pt\relax
8142     \gls@tmplen=0pt\relax
8143     #2=0pt\relax
8144     #3=0pt\relax
8145     \forallglossaries[#1]{\@gls@type}%
8146     {%
8147         \forglsentries[\@gls@type]{\@glo@label}%

```

```

8148     {%
8149         \ifglsused{\@glo@label}%
8150     {%
8151         \settowidth{\dimen@}%
8152             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8153         \ifdim\dimen@>\gls@tmplen
8154             \gls@tmplen=\dimen@
8155             \eglssetwidest{\glsentryname{\@glo@label}}%
8156         \fi
8157         \settowidth{\dimen@}%
8158             {\glsentrysymbol{\@glo@label}}%
8159         \ifdim\dimen@>#2\relax
8160             #2=\dimen@
8161         \fi
8162         \settowidth{\dimen@}%
8163             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
8164         \ifdim\dimen@>#3\relax
8165             #3=\dimen@
8166         \fi
8167     }%
8168 }%
8169 }%
8170 }%
8171 }

```

**eSymbolLocation** Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

8172 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
8173     \dimen@=0pt\relax
8174     \gls@tmplen=0pt\relax
8175     #2=0pt\relax
8176     #3=0pt\relax
8177     \foralllglossaries[#1]{\@gls@type}%
8178     {%
8179         \forglsentries[\@gls@type]{\@glo@label}%
8180     {%
8181         \settowidth{\dimen@}%
8182             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8183         \ifdim\dimen@>\gls@tmplen
8184             \gls@tmplen=\dimen@
8185             \eglssetwidest{\glsentryname{\@glo@label}}%
8186         \fi
8187         \settowidth{\dimen@}%
8188             {\glsentrysymbol{\@glo@label}}%
8189         \ifdim\dimen@>#2\relax
8190             #2=\dimen@
8191         \fi
8192         \settowidth{\dimen@}%
8193             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
8194         \ifdim\dimen@>#3\relax

```

```

8195         #3=\dimen@
8196     \fi
8197 }%
8198 }%
8199 }

```

**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.

```

8200 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
8201     \dimen@=0pt\relax
8202     \gls@tmplen=0pt\relax
8203     #2=0pt\relax
8204     \forallglossaries[#1]{\@gls@type}%
8205     {%
8206         \forallglsentries[\@gls@type]{\@glo@label}%
8207         {%
8208             \ifglsused{\@glo@label}%
8209             {%
8210                 \settowidth{\dimen@}%
8211                     {\glstreenamfmt{\glsentryname{\@glo@label}}}%
8212                 \ifdim\dimen@>\gls@tmplen
8213                     \gls@tmplen=\dimen@
8214                     \eglssetwidest{\glsentryname{\@glo@label}}%
8215                 \fi
8216                 \settowidth{\dimen@}%
8217                     {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
8218                 \ifdim\dimen@>#2\relax
8219                     #2=\dimen@
8220                 \fi
8221             }%
8222         }%
8223     }%
8224 }%
8225 }

```

**AnyNameLocation** Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

8226 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
8227     \dimen@=0pt\relax
8228     \gls@tmplen=0pt\relax
8229     #2=0pt\relax
8230     \forallglossaries[#1]{\@gls@type}%
8231     {%
8232         \forallglsentries[\@gls@type]{\@glo@label}%
8233         {%
8234             \settowidth{\dimen@}%
8235                 {\glstreenamfmt{\glsentryname{\@glo@label}}}%
8236             \ifdim\dimen@>\gls@tmplen
8237                 \gls@tmplen=\dimen@

```

```

8238         \eglssetwidest{\glstryname{\@glo@label}}}%
8239     \fi
8240     \settowidth{\dimen@}%
8241     {\GlsXtrFormatLocationList{\glstrynumberlist{\@glo@label}}}%
8242     \ifdim\dimen@>#2\relax
8243         #2=\dimen@
8244     \fi
8245 }%
8246 }%
8247 }

```

**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`.

```

8248 \newcommand*{\glsxtrComputeTreeIndent}[1]{%
8249 \glstreeindent=\glsxtrtreetopindent\relax
8250 }

```

**computeTreeSubIndent** `\glsxtrComputeTreeSubIndent{<level>}{<label>}{<register>}`

Compute the indent for the sub-entries. The first argument is the level, the second argument is the entry label and the third argument is the length register used to store the computed indent.

```

8251 \newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
8252 \ifcsundef{@glswidestname\romannumeral#1}%
8253 {%
8254 \settowidth{#3}{\glstreenamfmt{\@glswidestname\space}}%
8255 }%
8256 {%
8257 \settowidth{#3}{\glstreenamfmt{%
8258 \csname @glswidestname\romannumeral#1\endcsname\space}}%
8259 }%
8260 }

```

**treeSetHangIndent** Set `\hangindent` for top-level entries:

```

8261 \newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}

```

**treeSetSubHangIndent** Set `\hangindent` for sub-entries:

```

8262 \newcommand*{\glsxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}

```

Redefine `alttree`:

```

8263 \renewglossarystyle{alttree}{%
8264 \renewenvironment{theglossary}%
8265 {%
8266 \glsxtralttreeInit

```

```

8267     \def\@gls@prevlevel{-1}%
8268     \mbox{}\par}%
8269     {\par}%
8270 \renewcommand*\glossaryheader{}%
8271 \renewcommand*\glsgroupheading[1]{}%
8272 \renewcommand\glossentry[2]{%
8273     \ifnum\@gls@prevlevel=0\relax
8274     \else
8275         \glstrComputeTreeIndent{##1}%
8276     \fi
8277     \parindent\glstreeindent
8278     \glstrAltTreeSetHangIndent
8279     \makebox[Opt][r]%
8280     {%
8281         \glstreenamebox{\glstreeindent}%
8282         {%
8283             \glstryitem{##1}%
8284             \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8285         }%
8286     }%
8287     \glstralttreeSymbolDescLocation{##1}{##2}%
8288     \def\@gls@prevlevel{0}%
8289 }
8290 \renewcommand\subglossentry[3]{%
8291     \ifnum##1=1\relax
8292         \glssubentryitem{##2}%
8293     \fi
8294     \ifnum\@gls@prevlevel=##1\relax
8295     \else
8296         \glstrComputeTreeSubIndent{##1}{##2}{\gls@tmplen}%
8297         \ifnum\@gls@prevlevel<##1\relax
8298             \setlength\glstreeindent\gls@tmplen
8299             \addtolength\glstreeindent\parindent
8300             \parindent\glstreeindent
8301         \else
8302             \ifnum\@gls@prevlevel=0\relax
8303                 \glstrComputeTreeIndent{##2}%
8304             \else
8305                 \glstrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
8306             \fi
8307             \addtolength\parindent{-\glstreeindent}%
8308             \setlength\glstreeindent\parindent
8309         \fi
8310     \fi
8311     \glstrAltTreeSetSubHangIndent{##1}%
8312     \makebox[Opt][r]{\glstreenamebox{\gls@tmplen}{%
8313         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
8314     \glstralttreeSubSymbolDescLocation{##1}{##2}{##3}%
8315     \def\@gls@prevlevel{##1}%

```

```

8316     }%
8317     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8318   }
8319 }%
8320 {%

```

Assume the style isn't required if it hasn't already been defined.

```

8321 }

```

Reset the default style

```

8322 \ifx\@glossary@default@style\relax
8323 \else
8324   \setglossarystyle{\@glxtr@current@style}
8325 \fi

```



# Glossary

**First use** The first time a glossary entry is used (from the start of the document or after a reset) with one of the following commands: `\gls`, `\Gls`, `\GLS`, `\glspl`, `\Glspl`, `\GLSpl` or `\glsdisp`. *see* **first use flag** & **first use text**

**First use flag** A conditional that determines whether or not the entry has been used according to the rules of **first use**.

**First use text** The text that is displayed on **first use**, which is governed by the first and first-plural keys of `\newglossaryentry`. (May be overridden by `\glsdisp`.)

`makeindex` An indexing application.

`xindy` An flexible indexing application with multilingual support written in Perl.



\glsaccesssshortpl: new .....	121	\cGLSpl: new .....	83
\glsaccessssymbol: new .....	118	\cGLSpl@: new .....	83
\glsaccessssymbolplural: new .....	119	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new .....	116	new .....	78
\glstentryfmt: added check for short ..	44	\cGLS: new .....	83
\glslongpltok: new .....	150	\cGLSformat: new .....	83
\glsshortpltok: new .....	150	\cGLSpl: new .....	83
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new .....	83
name in \setkeys .....	151	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new .....	12
for plural .....	147	\glsenableentrycount: new .....	78
\GLSxtrlongpl: new .....	163	\glsfirstabbrvdefaultfont: new ..	154
\Glsxtrlongpl: new .....	163	\glsfirstlongdefaultfont: new ...	154
\glsxtrlongpl: new .....	162	\Glsfmtfirst: new .....	218
\glsxtrNoGlossaryWarning: new ....	15	\glsfmtfirst: new .....	217
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new .....	218
new .....	146	\glsfmtfirstpl: new .....	218
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new .....	217
new .....	146	\glsfmtplural: new .....	217
\glsxtrpostlinkendsentence: new ..	146	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new .....	162	\Glsxtrtitleshort .....	216
\Glsxtrshortpl: new .....	161	renamed from \glstentryfmtshort ..	216
\glsxtrshortpl: new .....	161	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort .....	215
\glslabeltok .....	172	renamed from \glstentryfmtshort ..	215
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok .....	170	\Glsxtrtitleshortpl .....	216
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl .....	216
redefinition of \acronymtype .....	12	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl .....	215
\Glsxtrshort .....	216	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl .....	215
\glsxtrshort .....	215	\Glsfmttext: new .....	217
\Glsfmtshortpl: changed to use		\glsfmttext: new .....	216
\glsxtrshortpl .....	216	\glshasattribute: new .....	127
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	127
\glsxtrshortpl .....	215	\GlsXtrEnableEntryCounting: new ..	78
\glsxtrifemptyglossary: new .....	18	\glsxtrifcounttrigger: new .....	80
\glsxtrnewnumber: added extra		\glsxtrscfont: new .....	182
argument .....	131	\glsxtrscsuffix: new .....	182
\glsxtrnewsymbol: added extra		\glsxtrsmfont: new .....	186
argument .....	130	\glsxtrsmsuffix: new .....	186
\MakeAcronymsAbbreviations: set the		short-em: new .....	193
default type to \acronymtype .....	91	short-em-desc: new .....	193
\newterm: fixed name argument .....	130	short-em-footnote: new .....	195
0.5 (2015-12-07)		short-em-long: new .....	191
\cGLS: new .....	83	short-em-long-desc: new .....	191
\cGLS@: new .....	83	short-em-postfootnote: new .....	195

short-sc-footnote: new .....	185	\Glsxtrheadtext: now uses headuc	
short-sc-postfootnote: new .....	185	attribute .....	209
short-sm: new .....	187	\glxtrheadtext: now uses headuc	
short-sm-desc: new .....	188	attribute .....	208
short-sm-footnote: new .....	189	short-long: switch off regular attribute	
short-sm-long: new .....	187	if set .....	171
short-sm-long-desc: new .....	187	short-long-desc: switch off regular	
short-sm-postfootnote: new .....	189	attribute if set .....	173
long-noshort-em: new .....	193	long-short: switch off regular attribute	
long-noshort-em-desc: new .....	194	if set .....	170
long-noshort-sm: new .....	188	long-short-desc: switch off regular	
long-noshort-sm-desc: new .....	188	attribute if set .....	171
long-short-em: new .....	190	footnote: switch off regular attribute if	
long-short-em-desc: new .....	190	set .....	174
long-short-sm: new .....	186	postfootnote: switch off regular	
long-short-sm-desc: new .....	186	attribute if set .....	175
0.5.1 (2015-12-02)		0.5.2 (2015-12-08)	
\Glsaccessstext: new .....	116	\@GLSdesc@: added accessibility support	53
0.5.1 (2015-12-07)		\@GLSdescplural@: added accessibility	
\@glxtr@doaccsupp: new .....	15	support .....	53
General: removed \ifglxtruseuhead	207	\@GLSfirst@: added accessibility	
\Glsaccessdesc: new .....	120	support .....	50
\Glsaccessdescplural: new .....	120	\@GLSfirstplural@: added accessibility	
\Glsaccessfirst: new .....	117	support .....	52
\Glsaccessfirstplural: new .....	118	\@GLSname@: added accessibility support	52
\Glsaccessname: new .....	116	\@GLSplural@: added accessibility	
\Glsaccessplural: new .....	117	support .....	51
\Glsaccesssymbol: new .....	118	\@GLSsymbol@: added accessibility	
\Glsaccesssymbolplural: new .....	119	support .....	54
\Glsxtrheadfirst: now uses headuc		\@GLSsymbolplural@: added	
attribute .....	210	accessibility support .....	54
\glxtrheadfirst: now uses headuc		\@GLStext@: added accessibility support	49
attribute .....	210	\@GLSdesc@: added accessibility support	53
\Glsxtrheadfirstplural: now uses		\@GLSdescplural@: added accessibility	
headuc attribute .....	211	support .....	53
\glxtrheadfirstplural: now uses		\@GLSfirst@: added accessibility	
headuc attribute .....	211	support .....	50
\Glsxtrheadplural: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute .....	210	support .....	52
\glxtrheadplural: now uses headuc		\@GLSname@: add accessibility support ..	52
attribute .....	209	\@GLsplural@: added accessibility	
\Glsxtrheadshort: now uses headuc		support .....	51
attribute .....	208	\@GLSsymbol@: added accessibility	
\glxtrheadshort: now uses headuc		support .....	54
attribute .....	207	\@GLSsymbolplural@: added	
\Glsxtrheadshortpl: now uses headuc		accessibility support .....	54
attribute .....	208	\@GLStext@: added accessibility support	50
\glxtrheadshortpl: now uses headuc		\@GLSdesc@: added accessibility support	53
attribute .....	207		

\@glsdescplural@: added accessibility support .....	53	\glsxtrnewabbrevpresetkeyhook: new .....	152
\@glsfirst@: added accessibility support .....	50	\glsxtrtagfont: new .....	144
\@glsfirstplural@: added accessibility support .....	51	\KV@printgloss@nonumberlist: added	44
\@glsname@: added accessibility support	52	\mfu@checkword@do: added .....	143
\@glsplural@: added accessibility support .....	51	\setabbreviationstyle: added check for post-definition style switch ....	166
\@glsymbol@: added accessibility support .....	54	0.5.3 (2015-12-09)	
\@glsymbolplural@: added accessibility support .....	54	\@glsxtr@autoindex@at: new .....	140
\@glstext@: added accessibility support	49	\@glsxtr@autoindex@encap: new ...	140
\@glsxtr@activate@initialtagging: new .....	144	\@glsxtr@autoindex@esc: new .....	141
\@glsxtr@do@titlecaps@warn: new	144	\@glsxtr@autoindex@level: new ...	141
\@glsxtr@tag: new .....	144	\@glsxtr@autoindex@setname: new	139
General: fixed typo in glossaries-accsupp and tidied up code to use just one		\@glsxtr@doabbreviationsdef: new	12
\@ifpackageloaded .....	115	General: removed	
removed \glsxtrabbrvfmt .....	164	\GlsXtrNoGlsWarningNoAutoMakeMain .....	106
\glossaryentrynumbers: added .....	41	\glsdescwidth: added .....	41
\Glossentrydesc: added .....	142	\glspagelistwidth: added .....	41
\Glossentryname: added .....	136	\glsxtrdoautoindexname: new .....	138
\Glossentrysymbol: added .....	142	\glsxtrpostnamehook: new .....	137
\glossentrysymbol: added .....	142	\if@glsxtr@format@override: new	137
\GLSaccessdesc: new .....	120, 125	\ProvidesGlossariesExtraLang: new	221
\GLSaccessdescplural: new ...	120, 125	\RequireGlossariesExtraLang: new	221
\GLSaccessfirst: new .....	117, 124	0.5.4 (2015-12-15)	
\GLSaccessfirstplural: new ..	118, 124	\@newglossaryentry@defunitcounters: new .....	84
\GLSaccesslong: new .....	122, 125	\@GLSxtr@p@acrlong@: new .....	71
\GLSaccesslongpl: new .....	122, 126	\@GLSxtr@p@acrlongpl@: new .....	72
\Glsaccesslongpl: new .....	122	\@GLSxtr@p@acrshort@: new .....	71
\glsaccesslongpl: new .....	122	\@GLSxtr@p@acrshortpl@: new .....	71
\GLSaccessname: new .....	116, 123	\@GLSxtr@p@long@: new .....	70
\GLSaccessplural: new .....	117, 123	\@GLSxtr@p@longpl@: new .....	71
\GLSaccessshort: new .....	121, 125	\@GLSxtr@p@plural@: new .....	69
\GLSaccessshortpl: new .....	122, 125	\@GLSxtr@p@short@: new .....	70
\GLSaccesssymbol: new .....	119, 124	\@GLSxtr@p@shortpl@: new .....	70
\GLSaccesssymbolplural: new	119, 124	\@GLSxtr@p@text@: new .....	69
\GLSaccessstext: new .....	116, 123	\@GlsXtrEnableOnTheFly: new .....	37
\glsentryfmt: moved		\@Glsxtr: new .....	38
\glssetabbrvfmt from		\@Glsxtr@p@acrlong@: new .....	71
\glsxtrabbrvfmt to here .....	44	\@Glsxtr@p@acrlongpl@: new .....	71
\GlsXtrEnableInitialTagging: new	143	\@Glsxtr@p@acrshort@: new .....	71
\glsxtrfieldtitlecase: new .....	131	\@Glsxtr@p@acrshortpl@: new .....	71
\GlsXtrFormatLocationList: new ...	42	\@Glsxtr@p@long@: new .....	70
		\@Glsxtr@p@longpl@: new .....	71
		\@Glsxtr@p@plural@: new .....	69
		\@Glsxtr@p@short@: new .....	70
		\@Glsxtr@p@shortpl@: new .....	70
		\@Glsxtr@p@text@: new .....	69

\@Glsxtrpl: new .....	39	\glxtr: new .....	38
\@alt@glshyp@opt: new .....	66	\glxtrcat: new .....	37
\@glscalt@hyp@opt: new .....	66	\glxtrdowrglossaryhook: new .....	66
\@glscalt@hyp@opt@char: new .....	66	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new .....	66	new .....	89
\@glsc@increment@currunitcount:		\GlsXtrEnableOnTheFly: new .....	37
new .....	85	\Glsxtrpl: new .....	39
\@glsc@local@increment@currunitcount:		\glxtrpl: new .....	38
new .....	85	\glxtrpostlocalreset: new .....	78
\@glsc@setdefault@glslink@opts:		\glxtrpostlocalunset: new .....	77
new .....	63	\glxtrpostreset: new .....	77
\@glxtr: new .....	38	\glxtrpostunset: new .....	77
\@glxtr@addunitcounter: new .....	84	\glxtrprotectlinks: new .....	68
\@glxtr@currunitcount: new .....	86	\GlsXtrSetAltModifier: new .....	66
\@glxtr@ifunitcounter: new .....	84	\GlsXtrSetDefaultGlsOpts: new ....	65
\@glxtr@p@acrlong@: new .....	71	\glxtrstarflywarn: new .....	37
\@glxtr@p@acrlongpl@: new .....	71	\GlsXtrWarning: new .....	39
\@glxtr@p@acrshort@: new .....	71	\MakeAcronymsAbbreviations: now	
\@glxtr@p@acrshortpl@: new .....	71	disables \setacronymstyle .....	91
\@glxtr@p@long@: new .....	70	1.0 (2016-01-24)	
\@glxtr@p@longpl@: new .....	71	\@glxtr@autoindexcrossrefs: new .	11
\@glxtr@p@plural@: new .....	69	\@glxtr@idx@displaynumberlist:	
\@glxtr@p@short@: new .....	69	new .....	98
\@glxtr@p@shortpl@: new .....	70	\@glxtr@idx@entrynumberlist: new	100
\@glxtr@p@text@: new .....	69	\@glxtr@noidx@displaynumberlist:	
\@glxtr@prevunitcount: new .....	86	new .....	98
\@glxtr@setentryunitcountunsetattr:		\@glxtr@noidx@entrynumberlist:	
new .....	90	new .....	99
\@glxtr@unitcountlist: new .....	84	\@glxtr@noidx@numberlistloop:	
\@glxtrpl: new .....	38	new .....	99
\@newglossaryentryposthook: added		\@glxtr@reg@glosslist: new .....	93
empty see value if not set and added		\makeglossaries: new .....	93
‘see’ to field key map .....	30	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new .....	37	\glxtrdiscardperiod: added check	
\cGlsformat: added .....	83	for first use .....	147
\cGlsformat: added .....	83	short-desc: fixed typo in	
\cGlsplformat: added .....	84	\glxtrinlinefullformat and	
\cGlsplformat: added .....	84	added missing second argument ...	178
\glsdisablehyper: added .....	68	1.02 (2016-04-25)	
\glsdohyperlink: added .....	67	\@glxtr@current@style: new .....	40
\glsdonohyperlink: added .....	68	\Glsfmtfull: new .....	220
\glsenableentryunitcount: new ....	86	\glsfmtfull: new .....	220
\glshasattribute: added check for		\Glsfmtfullpl: new .....	221
entry’s existence .....	127	\glsfmtfullpl: new .....	220
\glusifattribute: added check for		\Glsfmtlong: new .....	219
entry’s existence .....	128	\glsfmtlong: new .....	219
\glspostlinkhook: added existence		\Glsfmtlongpl: new .....	219
check .....	146	\glsfmtlongpl: new .....	219
\Glsxtr: new .....	38	\Glsxtrheadfull: new .....	214

\glxtrheadfull: new .....	213	\@GLSplural@: set abbreviation and regular format .....	51
\Glsxtrheadfullpl: new .....	215	\@GLSsymbol@: set regular format .....	54
\glxtrheadfullpl: new .....	214	\@GLSsymbolplural@: set regular format	54
\glxtrheadlong: new .....	212	\@GLStext@: set abbreviation and regular format .....	49
\Glsxtrheadlongpl: new .....	213	\@GLSuseri@: set regular format .....	55
\glxtrheadlongpl: new .....	212	\@GLSuserii@: set regular format .....	55
\Glsxtrtitlefull: new .....	214	\@GLSuseriii@: set regular format .....	55
\glxtrtitlefull: new .....	214	\@GLSuseriv@: set regular format .....	56
\Glsxtrtitlefullpl: new .....	215	\@GLSuserv@: set regular format .....	56
\glxtrtitlefullpl: new .....	214	\@GLSuservi@: set regular format .....	56
\Glsxtrtitlelong: new .....	213	\@Glsdesc@: set abbreviation and regular format .....	53
\glxtrtitlelong: new .....	212	\@Glsdescplural@: set abbreviation and regular format .....	53
\Glsxtrtitlelongpl: new .....	213	\@Glsfirst@: set abbreviation and regular format .....	50
\glxtrtitlelongpl: new .....	212	\@Glsfirstplural@: set abbreviation and regular format .....	52
\ifglxtrinsetinside: new .....	169	\@Glsname@: set abbreviation and regular format .....	52
postfootnote: added redef of \glxtrsetupfulldefs .....	176	\@Glsplural@: set abbreviation and regular format .....	51
stylemods: new .....	15	\@Glsymbol@: set regular format .....	54
1.03 (2016-04-27)		\@Glsymbolplural@: set regular format	54
\@GLSfirstplural@: bug fix: misspelt cs name .....	52	\@Glstext@: set abbreviation and regular format .....	50
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	51	\@GLSuseri@: set regular format .....	55
\@Glsfirstplural@: bug fix: misspelt cs name .....	52	\@GLSuserii@: set regular format .....	55
\@Glsplural@: fixed bug \@Glsplural@ should be redefined not \@Glsplural	51	\@GLSuseriii@: set regular format .....	55
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	51	\@GLSuseriv@: set regular format .....	55
\glxtrtitlelongpl: bug fix: changed \glxtrlong to \glxtrlongpl ..	212	\@GLSuserv@: set regular format .....	56
\glxtrtitleshortpl: bug fix: changed \glxtrshort to \glxtrshortpl	207	\@GLSuservi@: set regular format .....	56
1.04 (2015-04-30)		\@gls@preglossaryhook: added check for entry's existence .....	144
short-em-footnote: renamed from "footnote-em" .....	195	\@Glsdesc@: set abbreviation and regular format .....	53
1.04 (2016-05-02)		\@Glsdescplural@: set abbreviation and regular format .....	53
\@@glxtrpostloctag: new .....	43	\@Glsfirst@: set abbreviation and regular format .....	50
\@GLSdesc@: set abbreviation and regular format .....	53	\@Glsfirstplural@: set abbreviation and regular format .....	51
\@GLSdescplural@: set abbreviation and regular format .....	53	\@Glsname@: set abbreviation and regular format .....	52
\@GLSfirst@: set abbreviation format ..	50	\@Glsplural@: set abbreviation and regular format .....	51
\@GLSfirstplural@: set abbreviation and regular format .....	52	\@glsymbol@: set regular format .....	54
\@GLSname@: set abbreviation and regular format .....	52		

\@glssymbolplural@: set regular format	54	short-em-nolong-desc: new	193
\@glstext@: set abbreviation and regular format	49	short-em-postfootnote: renamed from “postfootnote-em”	195
\@glstr@deprecated@abbrstyle: new	168	short-footnote: new	175
\@glstr@do@style: new	16	short-long-user: new	202
\@glstr@doloctag: new	44	short-long-user-desc: new	203
\@glstr@idx@entrynumberlist: switched from \let to \newcommand	100	short-nolong: new	178
\@glstr@pagetag: new	43	short-nolong-desc: new	180
\@glstr@pagetag: new	43	short-postfootnote: new	177
\@glstr@preloctag: new	43	short-sc-footnote: renamed from “footnote-sc”	185
\@glstr@postloctag: new	43	short-sc-nolong: new	184
\@glstr@preloctag: new	43	short-sc-nolong-desc: new	184
\glossentrydesc: added glossdescfont attribute check	132	short-sc-postfootnote: renamed from “postfootnote-sc”	185
\Glossentryname: added glossnamefont attribute check	136	short-sm-footnote: renamed from “footnote-sm”	189
\glossentryname: added glossnamefont attribute check	134	short-sm-nolong: new	187
moved post name hook inside condition	136	short-sm-nolong-desc: new	188
\glsabbrvmfont: new	189	short-sm-postfootnote: renamed from “postfootnote-sm”	189
\glsabbrvuserfont: new	196	\letabbreviationstyle: new	168
\glsfirstabbrvmfont: new	189	\newabbreviationstyle: bug fix: corrected test for existence	167
\glsfirstabbrvuserfont: new	196	long-em-noshort-em: new	194
\glsfirstlongemfont: new	189	long-em-noshort-em-desc: new	195
\glsfirstlonguserfont: new	196	long-em-short-em: new	190
\glsifnotregularcategory: new	128	long-em-short-em-desc: new	191
\glslongdefaultfont: new	154	long-noshort: new	182
\glslongemfont: new	190	long-noshort-desc: new	181
\glslongfont: new	154	long-noshort-em: renamed from “long-em”	193
\glslonguserfont: new	196	long-noshort-em-desc: renamed from “long-desc-em”	194
\glstrassignfieldfont: new	49	long-noshort-sc: renamed from “long-sc”	184
\GlsXtrEnablePreLocationTag: new	42	long-noshort-sc-desc: renamed from “long-desc-sc”	185
\glstrfirstscfont: new	182	long-noshort-sm: renamed from “long-sm”	188
\glstrfirstsmfont: new	186	long-noshort-sm-desc: renamed from \long-desc-sm	188
\glstrlongshortdescsort: new	170	long-short-user: new	197
\glstrpostnamehook: added category check	137	long-short-user-desc: new	202
\glstrregularfont: new	44	\renewabbreviationstyle: new style: new	167
\glstruserfield: new	196		16
\glstruserparen: new	196		
\glstrusersuffix: new	196		
\GlsXtrWarnDeprecatedAbbrStyle: new	168		
short-em-long-em: new	192		
short-em-long-em-desc: new	192		
short-em-nolong: new	193		
		1.05 (2016-06-10)	
		\eglssetwidest: new	230
		\glsFindWidestAnyName: new	232



\glsFindWidestAnyNameLocation: new .....	237	\@GLSfirst@: added check for nohyperfirst attribute .....	51
\glsFindWidestAnyNameSymbol: new	235	\@GLSfirstplural@: added check for nohyperfirst attribute .....	52
\glsFindWidestAnyNameSymbolLocation: new .....	236	\@GLSxtrp: new .....	73
\glsFindWidestLevelTwo: new	233	\@Glsfirst@: added check for nohyperfirst attribute .....	50
\glsFindWidestUsedAnyName: new	231	\@Glsfirstplural@: added check for nohyperfirst attribute .....	52
\glsFindWidestUsedAnyNameLocation: new .....	237	\@Glsxtrp: new .....	72
\glsFindWidestUsedAnyNameSymbol: new .....	234	\@gls@preglossaryhook: added \glossxtrsetpopts .....	145
\glsFindWidestUsedAnyNameSymbolLocation: new .....	235	\@glsfirst@: added check for nohyperfirst attribute .....	50
\glsFindWidestUsedLevelTwo: new	232	\@glsfirstplural@: added check for nohyperfirst attribute .....	51
\glsFindWidestUsedTopLevelName: new .....	231	\@glsxtrinmark: new .....	205
\glsfirstlongfootnotefont: new	173	\@glsxtrnotinmark: new .....	205
\glsgetwidestname: new	230	\@glsxtrp: new .....	72
\glsgetwidestsubname: new	231	\@glsxtrp@opt: new .....	72
\glslongfootnotefont: new	173	\glossxtrsetpopts: new .....	72
\glsxtrAltTreeIndent: new	230	\glsps: new .....	74
\glsxtralttreeInit: new	230	\glspt: new .....	74
\glsxtrAltTreePar: new	230	\glsxtr@entry@p: new .....	73
\glsxtrAltTreeSetHangIndent: new	238	\glsxtrabbrvfootnote: new .....	173
\glsxtralttreeSubSymbolDescLocation: new .....	230	\glsxtrchecknohyperfirst: new	50
\glsxtralttreeSymbolDescLocation: new .....	229	\glsxtrfieldtitlecasecs: new	132
\glsxtrComputeTreeIndent: new	238	\glsxtrifinmark: new .....	205
\glsxtrComputeTreeSubIndent: new	238	\GLSxtrp: new .....	76
\glsxtrtreetopindent: new	230	\Glsxtrp: new .....	75
short-em-long: fixed incorrect font used by long form .....	191	\glsxtrp: new .....	73
\xglsssetwidest: new	230	\glsxtrsetpopts: new .....	72
1.06 (2016-06-18)		short-long-desc: added text key	173
\@glsdoifexistssorwarn: new	11	fixed misspelling of \glsabbrvfont in plural key .....	173
\@glsxtr@docdefval: new	10	long-short-desc: added missing text key .....	171
\@glsxtr@useseesee: new	30	fixed misspelling of \glsabbrvfont	171
General: disabled docdef key at the start of the document .....	18	footnote: changed first forms to use \glsfirstlongfootnotefont	173
docdef option changed to choice	10	postfootnote: removed \footnote from first keys .....	175
\glsxtr@useseesee: new	30	switched from \glsfirstlongfont to \glsfirstlongfootnotefont	176
\glsxtruseseesee: new	30	\RestoreAcronyms: modified \@gls@link@checkfirsthyper to set \glsxtrifwasfirstuse .....	92
\glsxtruseseeseeformat: new	31		
\if@glsxtrdocdefrestricted: new	11		
1.07 (2016-08-15)		1.08 (2016-12-13)	
\@@glsxtrp: new .....	72	\@@glsxtr@record: new .....	6

\@GLS@: added \@glxtr@record	46	\@glxtr@redef@forglseentries: new	5
\@GLSpl@: added \@glxtr@record	46	\@glxtr@shortcutsval: new	14
\@Gls@: added \@glxtr@record	45	\@glxtr@unsrt@getgrouptitle: new	114
\@GLspl@: added \@glxtr@record	46	\@print@noidx@glossary: added	
\@Gls@: added \@glxtr@record	45	redefinition	101
\@Gls@@link@: added		\glxtr@addloclistfield: added	
\@glxtr@record	46	group key	9
\@Gls@field@link: added		added location key	9
\@glxtr@record	45	\glxtr@fields: new	109
\@Gls@saveentrycounter: new	18	\glxtr@linkprefix: new	109
\@Glsdisp: added \@glxtr@record	46	\glxtr@org@newignoredglossary:	
\@GLspl@: added \@glxtr@record	45	new	25
\@glxtr@dorecord: new	7	\glxtr@s@newignoredglossary: new	26
\@glxtr@err@undefaction: new	5	\glxtr@shortcutsval: new	109
\@glxtr@record: new	6	\glxtr@texencoding: new	109
\@glxtr@warn@onexistsordo: new	5	\glxtr@writefields: new	109
\@glxtr@warn@undefaction: new	5	\GlsXtrLoadResources: new	109
\@print@unsrt@glossary: new	112	\glxtrresourcefile: changed	
General: added record package option	9	extension to .glstex	108
\glsadd: added \@glxtr@record	48	\newignoredglossary: added starred	
\glsdoifexists: now defines		version	25
\glslabel	28	1.12 (2017-02-03)	
\glxtr@do@wrglossary: new	18	\@@glxtr@recordcounter: new	7
\glxtr@addloclistfield: new	8	\@Gls@preglossaryhook: check for	
\glxtr@indexonly@saveentrycounter:		definition	145
new	8	\@glxtr@counterrecordhook: new	111
\glxtr@record: new	111	\@glxtr@display@loc: new	102
\glxtr@resource: new	109	\@glxtr@docounterrecord: new	111
\glxtr@saveentrycounter: new	18	\@glxtr@longnewglossaryentry:	
\glxtr@setup@record: new	8	new	25
\glxtrassignfieldfont: added check		\@glxtr@noop@recordcounter: new	7
for existence	49	\@glxtr@op@recordcounter: new	8
\glxtrresourcefile: new	108	\@glxtr@provide@storagekey: new	19
\printunsrtglossaries: new	112	\@glxtr@s@longnewglossaryentry:	
\printunsrtglossary: new	111	new	24
1.09 (2016-12-16)		\@glxtrentryfmt: new	20
\@glxtr@gettype: new	98	\@glxtrindexaliased: new	64
\@glxtr@mixed@assign@sortkey:		\@glxtrsetaliasnoindex: new	64
new	98	\@newglossaryentryposthook: added	
\@printglossary: redefined to save		check for alias key	34
options	97	\@no@glxtrindexaliased: new	64
\glxtr@makeglossaries: new	98	\@printunsrtglossary: new	112
1.10 (2016-12-17)		General: added target key to printgloss	
\@GLSpl@: fixed bug caused by typo in		family	97
command name	46	\apptoglossarypreamble: new	23
1.11 (2017-01-19)		\csGlsXtrLetField: new	22
\@glxtr@do@redef@forglseentries:		\eGlsXtrSetField: new	23
new	5	\gGlsXtrSetField: new	23
\@glxtr@noidx@do: new	114		

\glsdohyperlink: added check for alias field .....	67	\glsxtrresourcefile: added catcode change for @ .....	108
\glsnoidxdisplayloc: added redefinition .....	102	\glsxtrsetaliasnoindex: new .....	64
\glssettoctitle: added patch .....	26	\GlsXtrSetField: new .....	22
\glsxtr@counterrecord: new .....	111	\glsxtrsetfieldifexists: new .....	22
\glsxtr@langtag: new .....	109	\glsxtrunsrtdo: new .....	114
\glsxtr@newabbreviation: new ....	150	\Glsxtrusefield: new .....	22
\glsxtr@org@newignoredglossary: Added check for existence .....	25	\glsxtrusefield: new .....	22
\glsxtr@pluralsuffixes: new .....	109	short-postlong-user: new .....	200
\glsxtr@provideignoredglossary: new .....	27	short-postlong-user-desc: new ...	201
\glsxtr@s@newignoredglossary: Added check for existence .....	26	\longnewglossaryentry: added starred version .....	24
\glsxtr@s@provideignoredglossary: new .....	27	long-postshort-user: new .....	198
\glsxtrabbrvpluralsuffix: new ...	154	long-postshort-user-desc: new ...	199
\glsxtralias: new .....	34	postdot: new .....	12
\glsxtrcopytoglossary: new .....	28	\pretoglossarypreamble: new .....	24
\glsxtrdeffield: new .....	22	\print@noop@unsrtglossaryunit: new .....	113
\glsxtrdisplayendloc: new .....	102	\print@op@unsrtglossaryunit: new	113
\glsxtrdisplayendloohook: new ...	103	\printunsrtglossary: added starred form .....	111
\glsxtrdisplaysingleloc: new ....	102	\printunsrtglossaryhandler: new .	113
\glsxtrdisplaystartloc: new ....	102	\printunsrtglossaryunit: new .....	8
\glsxtrdeffield: new .....	22	\printunsrtglossaryunitsetup: new	113
\glsxtrentryfmt: new .....	20	\provideignoredglossary: new .....	27
\glsxtrfielddolistloop: new .....	21	\s@glsxtr@provide@storagekey: new	19
\glsxtrfieldforlistloop: new .....	21	\s@printunsrtglossary: new .....	112
\glsxtrfieldifinlist: new .....	22	\xGlsXtrSetField: new .....	23
\glsxtrfieldlistadd: new .....	21	1.13 (2017-02-07)	
\glsxtrfieldliststeadd: new .....	21	\@glsdisp: removed	
\glsxtrfieldlistgadd: new .....	21	\@glsxtr@org@glsdisp .....	46
\glsxtrfieldlistxadd: new .....	21	\glsxtrsetaliasnoindex: switched to \providecommand .....	64
\glsxtrfieldxifinlist: new .....	22	1.14 (2017-04-18)	
\glsxtrfmt: new .....	20	\@gls@link: added redefinition .....	47
\GlsXtrFmtDefaultOptions: new ....	20	\@gls@noidx@getgrouptitle: new ..	100
\GlsXtrFmtField: new .....	20	\@gls@removespaces: new .....	103
\glsxtrifkeydefined: new .....	19	\@glsxtr@do@automake@err: new ...	111
\glsxtrindexaliased: new .....	65	\@glsxtr@org@gloautosee: new .....	17
\GlsXtrLetField: new .....	22	\@glsxtr@record: added third arg .....	6
\GlsXtrLetFieldToField: new .....	23	\@glsxtr@recordsee: new .....	8
\GlsXtrLoadResources: removed restriction on only one per document	109	General: added \glsadd option theHvalue .....	48
\glsxtrlocrangefmt: new .....	103	added \glsadd option thevalue ....	48
\glsxtrpostlongdescription: new ..	25	\glsdisablehyper: added redefinition .	67
\glsxtrprovidestoragekey: new ....	19	\glsenableentrycount: fixed assignment of \@cGls@ .....	79
\GlsXtrRecordCounter: new .....	111	\glsenableentryunitcount: fixed assignment of \@cGls@ .....	88
\glsxtrresourcecount: new .....	109		

\glsnavigation: new .....	101	long-short-user: fixed spelling of	
\glsxtr@org@getgrouptitle: new ..	100	\glsabbrvfont .....	197
\glsxtr@recordsee: new .....	6	footnote: fixed spelling of	
\glsxtr@writefields: added check for		\glsabbrvfont .....	174
automake .....	110	postfootnote: fixed spelling of	
\glsxtrdisplayendloc: added check		\glsabbrvfont .....	175
for empty format .....	102	1.16 (2017-06-15)	
\glsxtrgetgrouptitle: new .....	100	\@glo@autosee: added redefinition ....	17
\glsxtrinitwrgloss: new .....	47	\@gls@noidx@getgrouptitle: fixed	
\glsxtrlocationhyperlink: new ...	103	bug .....	100
\glsxtrsetgrouptitle: new .....	100	\@glsxtr@addunusedxrefs: added	
\glsxtrsupphypernumber: new .....	103	check for seealso field .....	35
\ifglsxtrwrglossbefore: new .....	47	\@glsxtr@dorecordnodefer: new .....	7
1.15 (2017-05-10)		\@glsxtr@noidx@do: use \csuse instead	
\@glsxtr@dorecord: corrected		of \csname .....	114
premature expansion of \@glslocref	7	\@print@unsrt@glossary: corrected	
short-em-long-em: fixed spelling of		misspelt command .....	112
\glsabbrvfont .....	192	\@printunsrt@glossary@handler:	
short-long: fixed spelling of		new .....	113
\glsabbrvfont .....	171	General: added check for	
short-long-user: fixed spelling of		\@gls@setupsort@none .....	10
\glsabbrvfont .....	203	\gls@checkseeallowed: added	
short-postlong-user: fixed spelling of		redefinition .....	17
\glsabbrvfont .....	200	\glsxtr@writefields: added	
short-postlong-user-desc: fixed		\providecommand lines .....	110
spelling of \glsabbrvfont .....	202	\glsxtrautoindex: new .....	139
long-em-short-em: fixed spelling of		\glsxtrautoindexentry: new .....	139
\glsabbrvfont .....	190	\glsxtrautoindexsort: new .....	139
long-postshort-user: fixed spelling of		\glsxtrindexseealso: new .....	31
\glsabbrvfont .....	198	\glsxtrseealsolabels: new .....	34
long-postshort-user-desc: fixed		\glsxtrseelist: new .....	31
spelling of \glsabbrvfont .....	199	\glsxtruseseealso: new .....	31
long-short: fixed spelling of		\glsxtruseseealsoformat: new .....	31
\glsabbrvfont .....	169	\seealso: new .....	31
		autoseeindex: new .....	11

# 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	
\.	146, 229
\@	108
\@cGLS@	79, 88
\@cGLSpl@	79, 88
\@cGls@	79, 88
\@cGlspl@	79, 88
\@cgls@	79, 88
\@cglspl@	79, 81, 88
\@do@wrglossary	7, 94
\@do@wrglossary	9, 10, 18, 49, 64
\@glo@assign@sortkey	98
\@glo@list	5
\@glo@type	112
\@gls@expand@field	20
\@glslocalreset	77
\@glslocalunset	77
\@glsreset	77
\@glsunset	77
\@glsxtr@autoindex@escspch	140–142
\@glsxtr@checkspch	139, 140, 142
\@glsxtr@disabledflycommand	40
\@glsxtr@record	10
\@glsxtr@recordcounter	9, 10, 111
\@glsxtrp	72, 73
\@glsxtrpostloctag	42
\@glsxtrpreloctag	42, 43
\@newglossaryentry@defcounters	78
\@newglossaryentry@defunitcounters	86
\@par	230
\@ACRlong	69
\@ACRlongpl	69
\@ACRshort	69
\@ACRshortpl	69
\@Acrlong	69
\@Acrlongpl	69
\@Acrshort	69
\@Acrshortpl	69
\@GLS@	68, 82, 83
\@GLSdesc@	53
\@GLSpl@	68, 82, 83
\@GLSplural@	69
\@GLSsymbol@	54
\@GLStext@	69
\@GLSxtr@full	155
\@GLSxtr@fullpl	157
\@GLSxtrp@acrlong@	69
\@GLSxtrp@acrlongpl@	69
\@GLSxtrp@acrshort@	69
\@GLSxtrp@acrshortpl@	69
\@GLSxtrp@long@	68
\@GLSxtrp@longpl@	69
\@GLSxtrp@plural@	68
\@GLSxtrp@short@	68
\@GLSxtrp@shortpl@	69
\@GLSxtrp@text@	68
\@GLSxtrlong	68, 160
\@GLSxtrlongpl	69, 163
\@GLSxtrp	76, 77
\@GLSxtrshort	68, 158
\@GLSxtrshortpl	69, 162
\@Gls@	68, 81, 82
\@Gls@acentryname	90
\@Gls@entry@field	61, 75
\@Gls@entryname	90
\@GlsXtrEnableOnTheFly	37
\@Glspl@	68, 82
\@Glsplural@	69
\@Glstext@	69
\@Glsxtr	38, 39
\@Glsxtr@full	155
\@Glsxtr@fullpl	156
\@Glsxtrp@acrlong@	69
\@Glsxtrp@acrlongpl@	69

\@Glsxtr@p@acrshort@	69	\@glo@autosee	17
\@Glsxtr@p@acrshortpl@	69	\@glo@autoseehook	33
\@Glsxtr@p@long@	68	\@glo@category	84
\@Glsxtr@p@longpl@	69	\@glo@check@sortallowed	96
\@Glsxtr@p@plural@	68	\@glo@counterprefix	7, 103
\@Glsxtr@p@short@	68	\@glo@countunit	84
\@Glsxtr@p@shortpl@	69	\@glo@default@sorttype	96
\@Glsxtr@p@text@	68	\@glo@desc	24, 25
\@Glsxtrlong	68, 160	\@glo@descplural	24, 25
\@Glsxtrlongpl	69, 163	\@glo@group	9
\@Glsxtrp	75	\@glo@label	8, 9, 19, 30, 33–35, 61, 68, 231–238
\@Glsxtrpl	39	\@glo@location	9
\@Glsxtrshort	68, 158	\@glo@loclist	8
\@Glsxtrshortpl	69, 161	\@glo@name	138, 139
\@acrlong	69	\@glo@no@assign@sortkey	98
\@acrlongpl	69	\@glo@parent	232–234
\@acrshort	69	\@glo@see	30, 31, 33–35
\@acrshortpl	69	\@glo@seealso	32, 33
\@alt@glshyp@opt	66	\@glo@sort	139
\@auxout	7, 8, 43, 80, 88, 89, 93, 94, 104, 105, 108, 110, 111	\@glo@sorttype	96, 101
\@bibgls@restoreat	108, 109	\@glo@text	46
\@cGLS	83	\@glo@thislettergrp	114, 115
\@cGLS@	79, 83, 88	\@glo@thisvalue	196
\@cGLSpl	83	\@glo@tmp	19, 31, 61
\@cGLSpl@	79, 83, 88	\@glo@type	34, 90, 93, 96–98, 101, 102, 104, 105, 107, 108, 112
\@cGls@	79, 88	\@glo@types	129, 130, 231–237
\@cGlspl@	79, 88	\@glossary@default@style	40, 96, 240
\@cgls@	79, 88	\@glossarystyle	96, 97
\@cglspl@	79, 88	\@gls@	68, 81, 82
\@disable@onlypremakeg	94	\@gls@@link	46
\@do@auxoutstuff	104, 105	\@gls@ReturnAfterFi	103
\@do@glsee	33, 34	\@gls@actualchar	139
\@do@newglossaryentry	90, 91, 152	\@gls@adjustmode	48
\@do@seeglossary	9, 10, 16, 94	\@gls@alt@hyp@opt	66
\@do@wrglossary	48	\@gls@alt@hyp@opt@char	66
\@empty	49, 57–60, 139, 140, 154–164	\@gls@alt@hyp@opt@keys	66
\@end@glxtr@addunused	35	\@gls@automake	96
\@end@glxtr@gettype	96, 98	\@gls@between	101
\@end@glxtr@usesee	30	\@gls@checkedmkidx	139, 140, 142
\@endfortrue	166	\@gls@checkmkidxchars	32, 139
\@firstofone	49, 132, 133, 138, 144	\@gls@codepage	104, 105
\@firstofthree	46, 49, 57–60, 66, 154, 156, 158, 159, 161, 162	\@gls@counter	6, 7, 47, 48, 64
\@firstoftwo	50–54, 58, 60, 63, 66, 92, 147, 148, 155–157, 161–163, 205, 206	\@gls@currentlettergroup	101, 112, 114, 115
\@for	5, 15, 35, 78, 90, 93, 96, 101, 112, 131, 143	\@gls@declareoption	4
\@glo@alias	32, 33	\@gls@doautomake	96, 111
\@glo@assign@sortkey	96	\@gls@doautomake@err	111
		\@gls@encapchar	140
		\@gls@entry@count	80

<code>\@gls@entry@field</code>	19, 22, 33, 61, 73–76, 79	<code>\@gls@tmpb</code>	142
<code>\@gls@entry@unitcount</code>	88, 89	<code>\@gls@type</code>	94, 96, 166, 231–237
<code>\@gls@field@font</code>	49–56	<code>\@gls@write@entrycounts</code>	80
<code>\@gls@field@link</code>	49–56, 61, 62	<code>\@gls@write@entryunitcounts</code>	88
<code>\@gls@getgrouptitle</code>	100, 112	<code>\@gls@write@entryunitcounts@do</code>	89
<code>\@gls@grptitle</code>	101	<code>\@gls@xref</code>	8, 32
<code>\@gls@hyp@opt</code>	61, 62, 66, 83, 154–163	<code>\@glsabbrv@current@abbreviation</code>	151, 164
<code>\@gls@hyp@opt@cs</code>	66	<code>\@glsacronymlists</code>	90
<code>\@gls@increment@currcount</code>	79	<code>\@glsdoifexistsorwarn</code>	11, 134–137
<code>\@gls@increment@currunitcount</code>	87	<code>\@glsentry</code>	80, 88, 89
<code>\@gls@keymap</code>	8, 9, 19, 30, 32, 61, 110	<code>\@glslink</code>	48, 68
<code>\@gls@label</code>	7, 65, 66, 94, 111, 166	<code>\@glslocref</code>	7
<code>\@gls@levelchar</code>	139	<code>\@glsnextpages</code>	97
<code>\@gls@link</code>	20, 45–47, 57–61, 155–164	<code>\@glsnonextpages</code>	97
<code>\@gls@link@checkfirsthyper</code>	46, 92	<code>\@glsnumberformat</code>	6, 7, 47, 48, 64, 137, 138
<code>\@gls@link@label</code>	47	<code>\@glsorder</code>	93
<code>\@gls@link@nocheckfirsthyper</code>	45, 57–60, 154–163	<code>\@glspl@</code>	68, 81, 82
<code>\@gls@link@opts</code>	47	<code>\@glsplural@</code>	69
<code>\@gls@list</code>	101	<code>\@glspl@token</code>	148
<code>\@gls@local@increment@currcount</code>	79	<code>\@glsstyle@alttree</code>	229
<code>\@gls@local@increment@currunitcount</code>	87	<code>\@glsstyle@inline</code>	229
<code>\@gls@location</code>	114, 115	<code>\@glsstyle@listdotted</code>	224
<code>\@gls@loclist</code>	98, 99, 114, 115	<code>\@gls@target</code>	68, 97
<code>\@gls@longpl</code>	149, 151, 152	<code>\@gls@text@</code>	69
<code>\@gls@nohyperlist</code>	26, 27	<code>\@glswidestname</code>	230, 231, 238
<code>\@gls@noidx@do</code>	101	<code>\@glsxtr</code>	38, 39
<code>\@gls@noidx@getgrouptitle</code>	112	<code>\@glsxtr@@do@@wrglossary</code>	94
<code>\@gls@noidx@nosanitizesort</code>	95	<code>\@glsxtr@abbreviationsdef</code>	12, 17, 18
<code>\@gls@noidx@sanitizesort</code>	95	<code>\@glsxtr@activate@initialtagging</code>	143, 145
<code>\@gls@noidx@loclist@finalsep</code>	98	<code>\@glsxtr@addunitcounter</code>	84
<code>\@gls@noidx@loclist@prev</code>	98	<code>\@glsxtr@addunused</code>	35
<code>\@gls@noidx@loclist@sep</code>	98	<code>\@glsxtr@addunusedxrefs</code>	35
<code>\@gls@noref@warn</code>	94, 102	<code>\@glsxtr@attrval</code>	132–138
<code>\@gls@org@glsnoidx@displayloc</code>	99	<code>\@glsxtr@autoindex@at</code>	139, 140
<code>\@gls@org@glsseeformat</code>	99	<code>\@glsxtr@autoindex@doextra@esc</code>	139
<code>\@gls@preglossaryhook</code>	97, 143	<code>\@glsxtr@autoindex@encap</code>	138, 140
<code>\@gls@prevlevel</code>	239	<code>\@glsxtr@autoindex@esc</code>	139, 141, 142
<code>\@gls@quotechar</code>	139	<code>\@glsxtr@autoindex@escat</code>	139, 140
<code>\@gls@reference</code>	36, 93, 94	<code>\@glsxtr@autoindex@escencap</code>	140
<code>\@gls@saveentrycounter</code>	9, 10, 18, 48	<code>\@glsxtr@autoindex@esclevel</code>	140, 141
<code>\@gls@see@noindex</code>	17, 108, 109	<code>\@glsxtr@autoindex@escquote</code>	139, 141
<code>\@gls@setdefault@glslink@opts</code>	48, 65	<code>\@glsxtr@autoindex@level</code>	139–141
<code>\@gls@setsort</code>	48	<code>\@glsxtr@autoindex@setname</code>	138
<code>\@gls@setupsort@none</code>	10	<code>\@glsxtr@autoindex@crossrefs</code>	10, 11, 30, 33
<code>\@gls@short</code>	151	<code>\@glsxtr@autoseeindexfalse</code>	10
<code>\@gls@shortpl</code>	149, 151, 152	<code>\@glsxtr@autoseeindextrue</code>	11
<code>\@gls@sort</code>	114	<code>\@glsxtr@cat</code>	78, 90, 143
<code>\@gls@tmp</code>	101	<code>\@glsxtr@counterrecordhook</code>	7, 8

<code>\@glsxtr@csname</code> .....	85–87	<code>\@glsxtr@idx@entrynumberlist</code> .....	95
<code>\@glsxtr@current@style</code> .....	40, 240	<code>\@glsxtr@ifcsstart</code> .....	37
<code>\@glsxtr@currentunitcount</code> .....	85, 87	<code>\@glsxtr@ifpunctoken</code> .....	148
<code>\@glsxtr@currunitcount</code> .....	86, 88	<code>\@glsxtr@ifunitcounter</code> .....	84
<code>\@glsxtr@declareoption</code> .....	4, 12, 15	<code>\@glsxtr@insert@dots</code> .....	150
<code>\@glsxtr@defaultnoglossarywarning</code> ..	15	<code>\@glsxtr@insert@dots@next</code> .....	150
<code>\@glsxtr@deprecated@abbrstyle</code> .....	184–186, 188, 189, 194, 195	<code>\@glsxtr@insertdots</code> .....	151
<code>\@glsxtr@disabledflycommand</code> .....	39	<code>\@glsxtr@label</code> .....	35, 131
<code>\@glsxtr@display@loc</code> .....	102	<code>\@glsxtr@loadstyles</code> .....	223
<code>\@glsxtr@do@@wrindex</code> .....	65	<code>\@glsxtr@longnewglossaryentry</code> .....	24
<code>\@glsxtr@do@gl:disablehyperinlist</code> ..	63	<code>\@glsxtr@mixed@assign@sortkey</code> .....	96
<code>\@glsxtr@do@redef@forgl:entries</code> .....	6	<code>\@glsxtr@noidx@displaynumberlist</code> ...	95
<code>\@glsxtr@do@style</code> .....	16, 222	<code>\@glsxtr@noidx@do</code> .....	114
<code>\@glsxtr@do@titlecaps@warn</code> .	132–135, 144	<code>\@glsxtr@noidx@entrynumberlist</code> .....	95
<code>\@glsxtr@do@abbreviationsdef</code> .....	12	<code>\@glsxtr@noidx@numberlistloop</code> .....	95
<code>\@glsxtr@do@accsupp</code> .....	15, 16	<code>\@glsxtr@noop@recordcounter</code> .....	7, 9
<code>\@glsxtr@docdefval</code> .....	10, 11, 36	<code>\@glsxtr@notfoundinlist</code> .....	148
<code>\@glsxtr@docounterrecord</code> .....	8	<code>\@glsxtr@op@recordcounter</code> .....	10
<code>\@glsxtr@doglossary</code> .....	112, 113	<code>\@glsxtr@optlist</code> .....	39
<code>\@glsxtr@doloctag</code> .....	42, 43	<code>\@glsxtr@org@GLS@</code> .....	46
<code>\@glsxtr@dorecord</code> .....	7	<code>\@glsxtr@org@GLSpl@</code> .....	46
<code>\@glsxtr@dorecordnodefer</code> .....	7	<code>\@glsxtr@org@Gls@</code> .....	45
<code>\@glsxtr@dostylewarn</code> .....	166	<code>\@glsxtr@org@Glspl@</code> .....	46
<code>\@glsxtr@enabletagging</code> .....	143	<code>\@glsxtr@org@Glsxtr:titlefirst</code> .	205, 206
<code>\@glsxtr@end@</code> .....	37	<code>\@glsxtr@org@Glsxtr:titlefirstplural</code>	205, 206
<code>\@glsxtr@enddescpsh</code> .....	139–142	<code>\@glsxtr@org@Glsxtr:titlefull</code> ..	205, 207
<code>\@glsxtr@entrycount@org@localreset</code> .	79	<code>\@glsxtr@org@Glsxtr:titlefullpl</code>	205, 207
<code>\@glsxtr@entrycount@org@localunset</code> .	79	<code>\@glsxtr@org@Glsxtr:titlelong</code> ..	205, 206
<code>\@glsxtr@entrycount@org@reset</code> .....	79	<code>\@glsxtr@org@Glsxtr:titlelongpl</code>	205, 206
<code>\@glsxtr@entrycount@org@unset</code> .....	79	<code>\@glsxtr@org@Glsxtr:titleplural</code>	205, 206
<code>\@glsxtr@entryunitcount@org@localreset</code>	87	<code>\@glsxtr@org@Glsxtr:titleshort</code> .	205, 206
<code>\@glsxtr@entryunitcount@org@localunset</code>	87	<code>\@glsxtr@org@Glsxtr:titleshortpl</code>	205, 206
<code>\@glsxtr@entryunitcount@org@reset</code> ..	87	<code>\@glsxtr@org@Glsxtr:titletext</code> ..	205, 206
<code>\@glsxtr@entryunitcount@org@unset</code> ..	87	<code>\@glsxtr@org@MakeUppercase</code> ....	205, 206
<code>\@glsxtr@err@undefaction</code> .....	6, 9	<code>\@glsxtr@org@checkfirsthyper</code> ....	63, 92
<code>\@glsxtr@field@linkdefs</code> .....	45	<code>\@glsxtr@org@delimN</code> .....	43
<code>\@glsxtr@format@overridefalse</code> .....	137	<code>\@glsxtr@org@delimR</code> .....	43
<code>\@glsxtr@format@override&gt;true</code> .....	138	<code>\@glsxtr@org@doseeglossary</code> ....	9, 10, 94
<code>\@glsxtr@foundinlist</code> .....	148	<code>\@glsxtr@org@gloautosee</code> .....	17
<code>\@glsxtr@full</code> .....	154	<code>\@glsxtr@org@gls@</code> .....	45
<code>\@glsxtr@fullpl</code> .....	156	<code>\@glsxtr@org@glsignore</code> .....	43
<code>\@glsxtr@gettype</code> .....	96	<code>\@glsxtr@org@glspl@</code> .....	45
<code>\@glsxtr@glossdescfont</code> .....	132, 133	<code>\@glsxtr@org@glsxtr:titlefirst</code> .	205, 206
<code>\@glsxtr@glossnamefont</code> .....	134–137	<code>\@glsxtr@org@glsxtr:titlefirstplural</code>	205, 206
<code>\@glsxtr@gobbleto@enddescpsh</code> .....	142	<code>\@glsxtr@org@glsxtr:titlefull</code> ..	205, 206
<code>\@glsxtr@idx@displaynumberlist</code> .....	95	<code>\@glsxtr@org@glsxtr:titlefullpl</code>	205, 207
		<code>\@glsxtr@org@glsxtr:titlelong</code> ..	205, 206



<code>\@glxtr@org@glxtrtitlelongpl</code>	205, 206	<code>\@glxtr@tmp</code>	15, 16, 103
<code>\@glxtr@org@glxtrtitleplural</code>	205, 206	<code>\@glxtr@type</code>	131
<code>\@glxtr@org@glxtrtitleshort</code>	205, 206	<code>\@glxtr@unitcountlist</code>	84
<code>\@glxtr@org@glxtrtitleshortpl</code>	205, 206	<code>\@glxtr@unsrt@getgrouptitle</code>	112
<code>\@glxtr@org@glxtrtitletext</code>	205, 206	<code>\@glxtr@usesee</code>	30
<code>\@glxtr@org@makeglossaries</code>	93	<code>\@glxtr@warn@onexistsordo</code>	6, 10
<code>\@glxtr@org@markboth</code>	204, 205	<code>\@glxtr@warn@undefaction</code>	6, 10
<code>\@glxtr@org@markright</code>	204, 205	<code>\@glxtrdocdeffalse</code>	36
<code>\@glxtr@org@newacronymstyle</code>	91, 92	<code>\@glxtrentryfmt</code>	20
<code>\@glxtr@org@postdescription</code>	145	<code>\@glxtrindexaliased</code>	64
<code>\@glxtr@org@see@noindex</code>	108, 109	<code>\@glxtrindexcrossrefsfalse</code>	11
<code>\@glxtr@org@setacronymstyle</code>	91, 92	<code>\@glxtrindexcrossrefstrue</code>	11
<code>\@glxtr@org@printglossary</code>	39, 97	<code>\@glxtrinmark</code>	204
<code>\@glxtr@org@warndep</code>	149	<code>\@glxtrlong</code>	68, 159
<code>\@glxtr@p@acrlong@</code>	69	<code>\@glxtrlongpl</code>	69, 162
<code>\@glxtr@p@acrlongpl@</code>	69	<code>\@glxtrnotinmark</code>	204
<code>\@glxtr@p@acrshort@</code>	69	<code>\@glxtrp</code>	74
<code>\@glxtr@p@acrshortpl@</code>	69	<code>\@glxtrp@opt</code>	72
<code>\@glxtr@p@long@</code>	68	<code>\@glxtrpl</code>	38, 39
<code>\@glxtr@p@longpl@</code>	69	<code>\@glxtrpostloctag</code>	42, 44
<code>\@glxtr@p@plural@</code>	68	<code>\@glxtrpreloctag</code>	42, 44
<code>\@glxtr@p@short@</code>	68	<code>\@glxtrsetaliasnoindex</code>	64, 65
<code>\@glxtr@p@shortpl@</code>	69	<code>\@glxtrshort</code>	68, 157
<code>\@glxtr@p@text@</code>	68	<code>\@glxtrshortpl</code>	69, 161
<code>\@glxtr@pagetag</code>	42, 43	<code>\@glxtrundeftag</code>	5, 18
<code>\@glxtr@pagetag</code>	42, 43	<code>\@gobble</code>	6, 9, 10, 12, 49, 150
<code>\@glxtr@prevunitcount</code>	86	<code>\@gobbletwo</code>	149
<code>\@glxtr@printglossopts</code>	39, 96, 97	<code>\@ifnextchar</code>	66
<code>\@glxtr@provide@addstoragekey</code>	20	<code>\@ifpackageloaded</code>	...
<code>\@glxtr@provide@storagekey</code>	19		4, 12, 110, 115, 132, 133, 136, 138, 221
<code>\@glxtr@record</code>	9, 10, 45, 46, 48	<code>\@ifstar</code>	19, 24, 25, 27, 37, 66, 111, 143
<code>\@glxtr@recordsee</code>	10	<code>\@ifundefined</code>	221
<code>\@glxtr@redef@forglsentries</code>	6, 17, 18	<code>\@ignored@glossaries</code>	25–28
<code>\@glxtr@redefstyles</code>	15, 16, 222	<code>\@input</code>	109
<code>\@glxtr@reg@glosslist</code>	93–96, 98	<code>\@input@</code>	104
<code>\@glxtr@s@longnewglossaryentry</code>	24	<code>\@istfilename</code>	93
<code>\@glxtr@savepreloctag</code>	42, 43	<code>\@makeglossary</code>	93
<code>\@glxtr@setentrycountunsetattr</code>	78	<code>\@mfu@domakefirsttuc</code>	144
<code>\@glxtr@setentryunitcountunsetattr</code>	89, 90	<code>\@mfu@nocaplist</code>	144
<code>\@glxtr@setupshortcuts</code>	14, 15, 17, 18	<code>\@one</code>	80, 89
<code>\@glxtr@shortcutsval</code>	14, 110	<code>\@newglossaryentry@defcounters</code>	78, 86
<code>\@glxtr@swaptwo</code>	149	<code>\@newglossaryentryposthook</code>	8, 9, 19, 32, 61
<code>\@glxtr@tag</code>	143	<code>\@newglossaryentryprehook</code>	8, 9, 19, 24, 25, 32, 61
<code>\@glxtr@taggingcs</code>	143	<code>\@nil</code>	103, 114
<code>\@glxtr@theHvalue</code>	6, 7, 48, 49	<code>\@nnil</code>	139, 140, 142, 148, 150
<code>\@glxtr@thevalue</code>	6, 7, 48, 49	<code>\@no@glxtrindexaliased</code>	64, 65
<code>\@glxtr@thisloctag</code>	43	<code>\@no@makeglossaries</code>	108
<code>\@glxtr@titlelabel</code>	100, 101, 114	<code>\@nopostdesc</code>	97

\@onelevel@sanitize	8, 32, 39, 100, 101, 114	\ACRfullplfmt	91
\@onlypreamble	40, 43, 88, 109, 111, 138, 140, 141, 143	\Acrfullplfmt	91
\@org@glossaryentrynumbers	97	\acrfullplfmt	91
\@org@newglossaryentryprehook	24, 25	\acronymentry	91
\@print@unsrt@glossary	112	\acronymfont	57–60, 71, 91, 92
\@printgloss@setsort	96, 97	\acronymname	12
\@printglossary	39, 112	\acronymsort	91
\@printunsrt@glossary@handler	112	\acronymtype	12, 90, 91
\@printunsrtglossary	111	\acrpluralsuffix	91, 110
\@sGlsXtrEnableOnTheFly	37	\actualchar	141
\@secondofthree	50–52, 57–60, 62, 155, 157, 158, 160, 161, 163	\addtolength	239
\@secondoftwo	46, 49, 52–60, 63, 68, 92, 97, 149, 154–156, 158–163, 176, 205, 206	\advance	80, 89, 109
\@sglsxtr@provide@storagekey	19	\AF	13
\@thirdofthree	50–52, 57–60, 62, 156, 157, 159, 160, 162, 163	\Af	13
\@thirdoftwo	52–56	\af	13
\@warn@nomakeglossaries	105	\AFP	13
\@xdy@main@language	104	\Afp	13
\@xdycrossrefhook	31	\afp	13
\@xdy@language	104	\AL	13
\@xdylocationclassorder	32	\Al	13
\\	103	\al	13
\_	106, 107	\ALP	13
<b>A</b>		\Alp	13
\AB	13	\alp	13
\Ab	13	\AnyTrackedLanguages	221
\ab	13	\appto	8, 9, 16, 19, 30–34, 61, 65, 78, 86, 113, 138, 148, 150
abbreviation styles:		\AS	13
long-noshort	193, 194	\As	13
long-postshort-user	199	\as	13
long-short-user	198	\ASP	13
short	178	\Asp	13
short-long-user	200	\asp	13
short-postlong-user	201	\AtBeginDocument	18, 41, 110
\abbreviationsname	12	\AtEndDocument	34, 80, 88, 104
\abbrvpluralsuffix	110, 151, 170, 172, 174, 176, 177, 179, 180, 182–189, 197, 198, 200, 203	<b>B</b>	
\ABP	13	babel package	138, 140, 147
\Abp	13	\begin	101, 106, 112, 224–228
\abp	13	\begingroup	6, 64, 112
\ACRfullfmt	91	\bgroup	24, 25, 97
\Acrfullfmt	91	<b>C</b>	
\acrfullfmt	91	\catcode	108
		category attributes:	
		discardperiod	147
		entrycount	77, 78, 80, 89
		firstuc	136
		glossdesc	132
		glossdescfont	132

glossname	133		
glossnamefont	134, 136		
headuc	207		
indexname	139		
indexonlyfirst	65		
insertdots	151		
nohyper	63		
nohyperfirst	50–52		
regular	44, 83, 169–171, 173–175, 178, 179, 181, 190, 192, 197, 203		
\cGLS	13, 78, 89		
\cGls	13, 78, 89		
\cglS	13, 78, 89		
\cGLSformat	82		
\cGlsformat	81		
\cglSformat	81, 83		
\cGLSpl	13, 78, 89		
\cGlspl	13, 78, 89		
\cglSpl	13, 78, 89		
\cGLSplformat	82		
\cGlsplformat	82		
\cglSplformat	81, 83		
\char	100		
\columnwidth	41		
\count@	80, 89		
\csappto	23		
\csdef	19, 22–24, 61, 62, 79, 84, 85, 87, 126, 167, 168, 175, 198–200, 202, 224		
\cseappto	28		
\csedef	22, 86		
\csgdef	23, 25–28, 36, 42, 79, 80, 85, 87, 88		
\cslet	22, 24, 25, 97		
\csletcs	22, 23, 168		
\csname	5, 6, 20, 26, 27, 32, 40, 44, 46– 48, 57–62, 64, 72, 85, 86, 94, 101, 104, 107, 108, 112, 132, 149, 155–164, 169, 238		
\cspreto	24		
\csuse	20, 21, 26, 34, 42, 61, 62, 74, 75, 84–88, 96, 100–102, 111, 113, 114, 126, 137, 145, 146, 167, 168, 231–234		
\csxdef	30, 33, 85, 88, 101		
\currentglossary	97		
\CurrentOption	16, 223		
\CurrentTrackedLanguageTag	110		
\CurrentTrackedTag	221		
\CustomAbbreviationFields	152, 169, 171–173, 175, 177, 179–181, 190, 192, 194, 197–202		
		<b>D</b>	
\DeclareAcronymList	90		
\DeclareOption	4, 223		
\DeclareOptionX	4, 16		
\def	6–10, 18, 24–27, 30, 32, 35, 37–39, 41, 44–60, 64, 66, 68–72, 81–83, 90, 94, 96–98, 100–104, 112, 114, 137, 139–144, 148–151, 154–164, 166, 239		
\defglSentryfmt	26–28		
\define@boolkey	11, 63		
\define@choicekey	6, 9, 11, 14, 15, 47, 97		
\define@key	8, 9, 15, 16, 19, 32, 48, 61, 149		
\DefineAcronymSynonyms	14		
\delimN	43		
\delimR	43		
\detokenize	37		
\dimen@	92, 231–238		
\dimen@i	232–234		
\dimen@ii	232–234		
\dimexpr	41, 230		
\disable@keys	12, 18, 36		
\do	5, 15, 35, 78, 90, 93, 96, 101, 112, 131, 143		
\do@glS@link@checkfirsthyper	20, 45, 46, 48, 57–60, 154–163		
\do@glSdisablehyperinlist	48, 64		
doc package	141		
\dolistcsloop	21		
\DTLifinlist	94, 95, 98		
\DTLifint	100		
		<b>E</b>	
\eappto	8, 16, 25–28, 112, 138, 223		
\edef	5, 6, 25–28, 31, 33, 34, 47, 48, 63, 64, 68, 84, 85, 87, 93, 94, 98, 100, 102–104, 108, 132–137, 139, 140, 142, 149, 232–234		
\eglssetwidest	231–238		
\egroup	24, 25, 97		
\else	7, 8, 11, 12, 14, 15, 17, 23, 36, 37, 42, 44, 46, 48, 64, 65, 81, 92, 95, 97, 100, 102, 103, 106, 108, 109, 138–140, 142, 148, 150, 158–164, 170, 172, 174– 181, 197–199, 201, 203, 225–229, 239, 240		
\emph	189, 190		
\empty	102, 103		
\encapchar	141		
\end	101, 106, 107, 113, 224–228		
\end@glSxtr@display@loc	102		
\endcsname	5, 6, 20, 26, 27, 32, 40, 44, 46– 48, 57–62, 64, 72, 85, 86, 94, 101, 104, 107, 108, 112, 132, 149, 155–164, 169, 238		

<code>\endgroup</code> .....	7, 64, 112	<code>glossaries-extra-stylemods package</code> .	15, 145, 222
entry categories:		<code>glossaries.sty package</code> .....	25
abbreviation .....	164	<code>\GlossariesExtraWarning</code> .....	
general .....	126, 128	.....	5, 12, 23, 24, 37, 39, 92, 94,
index .....	130		103, 106, 109, 112, 132–137, 143, 144, 168
<code>\epreto</code> .....	139	<code>\GlossariesExtraWarningNoLine</code> .	12, 80, 89
<code>\equal</code> .....	107, 108	<code>\GlossariesWarning</code> .....	42, 94, 96, 99, 166
etoolbox package .....	4	<code>\GlossariesWarningNoLine</code> .....	94, 105
<code>\expandafter</code> .....	16,	glossary styles:	
20, 30, 31, 35, 37–39, 61, 62, 66, 72, 83,		alttree .....	229, 230, 238
84, 94–96, 98, 101–103, 112, 114, 115,		inline .....	229
132, 135, 136, 138, 141, 142, 148, 151, 152		listdotted .....	224
<code>\expandonce</code> .....	91, 139, 140	listdottedstyle .....	224
		sublistdotted .....	224
<b>F</b>		<code>glossary-long package</code> .....	225
<code>\fi</code> .....	6–8, 10–12, 14,	<code>glossary-longbooktabs package</code> .....	225
15, 17, 23, 30, 33, 34, 36, 37, 40–42, 44,		<code>glossary-mcols package</code> .....	223
46–48, 64, 65, 80, 81, 88, 89, 92, 95–98,		<code>glossary-tree package</code> .....	223
100, 102, 103, 105–109, 111, 138–140,		<code>\glossaryentrynumbers</code> ....	44, 97, 114, 115
142, 148, 150, 158–164, 170, 172, 174–		<code>\glossaryheader</code> ....	101, 112, 224–228, 239
181, 197–199, 201, 203, 225–229, 231–240		<code>\glossaryname</code> .....	96
first use .....	241	<code>\glossarypostamble</code> .....	101, 113
flag .....	241	<code>\glossarypreamble</code> .....	101, 112
text .....	241	<code>\glossarysection</code> ...	101, 102, 107, 112, 113
<code>\firstacronymfont</code> .....	91, 92	<code>\glossarytitle</code> 26, 27, 96, 97, 101, 102, 107, 112	
fontspec package .....	110	<code>\glossarytoctitle</code> .....	
<code>\footnote</code> .....	173	.....	26, 27, 96, 101, 102, 107, 112
<code>\forall glossaries</code> 34, 112, 129–131, 231–237		<code>\glossentry</code> .....	97, 115, 224–228, 239
<code>\forall glossentries</code> .....	80, 89	<code>\glossentrydesc</code> .....	224–230
<code>\ForEachTrackedDialect</code> .....	221	<code>\glossentryname</code> .....	224–228, 239
<code>\forall glossentries</code> ....	5, 34, 129–131, 231–237	<code>\glossentrysymbol</code> .....	225–230
<code>\forall listcsloop</code> .....	21, 89, 101	<code>\glossxtrsetpopts</code> .....	145
<code>\forall listloop</code> .....	98, 99, 144	<code>\GLS</code> .....	78, 89
<code>\futurelet</code> .....	148	<code>\Gls</code> .....	38, 78, 89
		<code>\gls</code> .....	23, 38, 40, 78, 89, 95, 105
<b>G</b>		<code>\gls@assign@desc</code> .....	24, 25
<code>\gdef</code> .....	43, 140, 141	<code>\gls@assign@field</code> .....	8, 9, 19, 61
<code>\Genacrfullformat</code> .....	91	<code>\gls@checkseeallowed</code> .....	36, 94
<code>\genacrfullformat</code> .....	91	<code>\gls@codepage</code> .....	105
<code>\GenericAcronymFields</code> .....	91	<code>\gls@defdocnewglossaryentry</code> ....	78, 86
<code>\Genplacrfullformat</code> .....	91	<code>\gls@defglossaryentry</code> ....	24, 25, 38, 39
<code>\genplacrfullformat</code> .....	91	<code>\gls@dotocitle</code> .....	97
<code>\glo@grabfirst</code> .....	114	<code>\gls@glossary</code> .....	32
<code>\glo@name</code> .....	134–136	<code>\gls@level</code> .....	114
<code>\gloaliaslabel</code> .....	67	<code>\gls@noidxglossary</code> .....	94
<code>\global</code> .....	24, 25, 97, 114, 115	<code>\gls@org@glossaryentryfield</code> .....	97
<code>\glo linkprefix</code> .....	48, 67, 68, 110, 113	<code>\gls@org@glossarysubentryfield</code> ....	97
<code>glossaries package</code> ....	10, 17, 30–32, 34, 96, 223	<code>\gls@save@numberlist</code> .....	41, 42, 44
<code>glossaries-accsupp package</code> .....	15, 16, 115	<code>\gls@set@xr@key</code> .....	32
<code>glossaries-extra package</code> .....	2		

<code>\gls@tmplen</code> .....	231–237, 239	<code>\Glsaccessstext</code> .....	50
<code>\gls@type</code> .....	94	<code>\glsaccessstext</code> .....	49
<code>\glsabbrvdefaultfont</code> .....		<code>\glsacrshortcutstrue</code> .....	14
.....	154, 170, 172, 174, 176, 177, 179, 180	<code>\glsacspacemax</code> .....	92
<code>\glsabbrvfont</code> .....	189–195	<code>\glsadd</code> .....	35, 106
<code>\glsabbrvfont</code> 69, 70, 91, 154, 158, 159, 161,		<code>\glsadd options</code>	
162, 164, 165, 169–177, 179–195, 197–203		thevalue .....	7
<code>\glsabbrvuserfont</code> .....	196–198, 200, 203	<code>\glsaddstoragekey</code> .....	34, 126
<code>\GLSaccessdesc</code> .....	53	<code>\glsbackslash</code> .....	37
<code>\Glsaccessdesc</code> .....	53, 132, 142	<code>\glsapscase</code> .....	46, 49–60, 62, 154–165
<code>\glsaccessdesc</code> .....	53, 133, 146	<code>\glsacategory</code> .....	
<code>\GLSaccessdescplural</code> .....	53	.....	44, 49, 63, 69, 70, 127–129, 132–
<code>\Glsaccessdescplural</code> .....	53	137, 142, 143, 145, 146, 154–159, 161, 162	
<code>\glsaccessdescplural</code> .....	53	<code>\glsacategorylabel</code> .....	
<code>\GLSaccessfirst</code> .....	51	.....	63, 149, 151, 175, 198–200, 202
<code>\Glsaccessfirst</code> .....	50	<code>\glsaclosebrace</code> .....	32, 106–108
<code>\glsaccessfirst</code> .....	50	<code>\glsacurrententrylabel</code> .....	
<code>\GLSaccessfirstplural</code> .....	52	.....	42, 43, 97, 104, 112, 113, 144, 145
<code>\Glsaccessfirstplural</code> .....	52	<code>\glsacurrentfieldvalue</code> .....	20, 21, 196
<code>\glsaccessfirstplural</code> .....	51	<code>\glsacustomtext</code> ..	45, 46, 57–60, 154–164, 166
<code>\Glsaccesslong</code> .....		<code>\glsadefaulttype</code> 5, 12, 23, 24, 96, 105, 112, 113	
.....	59, 153, 160, 170, 178, 181, 197, 199	<code>\glsadescriptionaccessdisplay</code> 119, 120, 132	
<code>\glsaccesslong</code> .....	59, 152, 159,	<code>\glsadescriptionpluralaccessdisplay</code> 120	
160, 170, 172, 174–181, 197–199, 201, 203		<code>\glsadescwidth</code> .....	224, 226–228
<code>\Glsaccesslongpl</code> .....		<code>\glsadetoklabel</code> .....	21–25,
.....	60, 153, 163, 170, 178, 181, 198, 199	28–31, 35–37, 47, 48, 64, 67, 79, 80, 85–	
<code>\glsaccesslongpl</code> .	60, 153, 163, 164, 170,	89, 94, 97–99, 114, 132, 134–136, 232–234	
172, 174, 175, 177–181, 197–199, 201, 203		<code>\glsadisplaynumberlist</code> .....	95, 98
<code>\GLSaccessname</code> .....	52	<code>\glsadohyperlink</code> .....	68
<code>\Glsaccessname</code> .....	52	<code>\glsadohypertarget</code> .....	97
<code>\glsaccessname</code> .....	52	<code>\glsadoifexists</code> .....	11,
<code>\GLSaccessplural</code> .....	51	22, 28, 30, 31, 45, 46, 48, 57–60, 99, 154–163	
<code>\Glsaccessplural</code> .....	51	<code>\glsadoifexistsordo</code> .....	20, 21, 46
<code>\glsaccessplural</code> .....	51	<code>\glsadoifexistsorwarn</code> ....	11, 132, 133, 142
<code>\Glsaccessshort</code> .....		<code>\glsadoifnoexists</code> .....	24, 25
. 57, 158, 165, 172, 174–177, 179, 201, 203		<code>\glsadonohyperlink</code> .....	48, 68
<code>\glsaccessshort</code> ..	57, 152, 153, 158, 159,	<code>\glsadosanitizesort</code> .....	95
165, 170, 172, 174–181, 197, 199, 201, 203		<code>\glsenableentrycount</code> .....	78, 80, 88
<code>\Glsaccessshortpl</code> .....		<code>\glsenableentryunitcount</code> .....	80, 89
. 58, 161, 164, 172, 174–177, 179, 201, 203		<code>\glsentrycounter</code> .....	103
<code>\glsaccessshortpl</code> ....	58, 153, 161, 162,	<code>\glsentrycurrcount</code> .....	79, 80, 86
164, 170, 172, 174–181, 197–199, 201, 203		<code>\Glsentrydesc</code> .....	120, 124, 133
<code>\GLSaccesssymbol</code> .....	54	<code>\glsentrydesc</code> .....	119, 120, 124, 125, 133
<code>\Glsaccesssymbol</code> .....	54, 143	<code>\Glsentrydescplural</code> .....	120, 125
<code>\glsaccesssymbol</code> .....	54, 142, 146	<code>\glsentrydescplural</code> .....	120, 125
<code>\GLSaccesssymbolplural</code> .....	54	<code>\Glsentryfirst</code> .....	83, 117, 123
<code>\Glsaccesssymbolplural</code> .....	54	<code>\glsentryfirst</code> ...	83, 117, 118, 123, 124, 218
<code>\glsaccesssymbolplural</code> .....	54	<code>\Glsentryfirstplural</code> .....	84, 118, 124
<code>\GLSaccessstext</code> .....	50	<code>\glsentryfirstplural</code> ....	84, 118, 124, 218

<code>\glstentryfmt</code> .....	26–28	<code>\glsfirstabbrvdefaultfont</code> .....	
<code>\Glsentryfull</code> .....	91	.....	154, 170, 172, 174, 176, 177, 179, 180
<code>\glstentryfull</code> .....	91	<code>\glsfirstabbrvemfont</code> .....	191–195
<code>\Glsentryfullpl</code> .....	91	<code>\glsfirstabbrvfont</code> .....	
<code>\glstentryfullpl</code> .....	91	.....	91, 152, 153, 169–195, 197–199, 201–203
<code>\glstentryitem</code> .....	224–228, 239	<code>\glsfirstabbrvuserfont</code> ..	197, 198, 200–203
<code>\Glsentrylong</code> .....	70, 71, 83, 122, 125	<code>\glsfirstaccessdisplay</code> .....	117, 118
<code>\glstentrylong</code> .....		<code>\glsfirstlongdefaultfont</code> .....	
...	70, 71, 83, 122, 125, 176, 200, 202, 219	.....	170, 172, 177, 179, 180
<code>\Glsentrylongpl</code> .....	71, 84, 122, 126	<code>\glsfirstlongemfont</code> .....	191–195
<code>\glstentrylongpl</code> ..	71, 72, 84, 122, 126, 219, 220	<code>\glsfirstlongfont</code> .....	152,
<code>\Glsentryname</code> .....	116, 123, 134–137	.....	153, 169–174, 176–181, 190–195, 197–203
<code>\glstentryname</code> ...	115, 116, 123, 139, 231–238	<code>\glsfirstlongfootnotefont</code> .....	173–177
<code>\glstentrynumberlist</code> .....	95, 100, 236–238	<code>\glsfirstlonguserfont</code> ..	197, 198, 201, 203
<code>\Glsentryplural</code> .....	117, 123	<code>\GLSfirstplural</code> .....	211
<code>\glstentryplural</code> .....	117, 123, 217	<code>\Glsfirstplural</code> .....	211, 212
<code>\glstentryprevcount</code> .....	79, 80, 86	<code>\glsfirstplural</code> .....	211
<code>\glstentryprevmaxcount</code> .....	87	<code>\glsfirstpluralaccessdisplay</code> .....	118
<code>\glstentryprevtotalcount</code> .....	87	<code>\glsforeachincategory</code> .....	166
<code>\Glsentryshort</code> .....	70, 71, 121, 125	<code>\glsgenentryfmt</code> .....	44
<code>\glstentryshort</code> .....		<code>\glsgetattribute</code> ..	67, 80, 85, 86, 104, 132–138
...	69–71, 92, 121, 125, 198, 200, 215, 216	<code>\glsgetcategoryattribute</code> .....	127
<code>\Glsentryshortpl</code> .....	70, 71, 121, 125	<code>\glsgetwidestname</code> .....	230
<code>\glstentryshortpl</code> .....		<code>\glsgroupheading</code> .....	115, 224–228, 239
.....	70, 71, 121, 122, 125, 215, 216	<code>\glsgroupskip</code> .....	115, 225–229, 240
<code>\Glsentrysymbol</code> .....	119, 124	<code>\glshasattribute</code> .....	67, 80,
<code>\glstentrysymbol</code> ....	118, 119, 124, 235, 236	.....	85, 87, 89, 104, 132–138, 170, 171, 173,
<code>\Glsentrysymbolplural</code> .....	119, 124	.....	174, 176, 190, 192, 197, 198, 200, 202, 203
<code>\glstentrysymbolplural</code> .....	119, 124	<code>\glshascategoryattribute</code> .....	127
<code>\Glsentrytext</code> .....	116, 123	<code>\glshyperlink</code> .....	68
<code>\glstentrytext</code> .....	68, 116, 123, 216, 217	<code>\glshypernavsep</code> .....	101
<code>\Glsentryuseri</code> .....	55	<code>\glshypernumber</code> .....	104, 138
<code>\glstentryuseri</code> .....	55	<code>\glsifattribute</code> .....	47, 50,
<code>\Glsentryuserii</code> .....	55	.....	63, 65, 73, 130, 132–135, 144, 147, 207–215
<code>\glstentryuserii</code> .....	55	<code>\glsifcategory</code> .....	129
<code>\Glsentryuseriii</code> .....	55	<code>\glsifcategoryattribute</code> .....	63, 128, 151
<code>\glstentryuseriii</code> .....	55	<code>\glsifnotregular</code> .....	49
<code>\Glsentryuseriv</code> .....	56	<code>\glsifnotregularcategory</code> .....	129
<code>\glstentryuseriv</code> .....	56	<code>\glsifplural</code> ..	46, 49, 51–54, 57–60, 147, 154–165
<code>\Glsentryuserv</code> .....	56	<code>\glsifregular</code> .....	44, 49, 83, 84
<code>\glstentryuserv</code> .....	56	<code>\glsifregularcategory</code> .....	129
<code>\Glsentryuservi</code> .....	56	<code>\glsifusetranslator</code> .....	26
<code>\glstentryuservi</code> .....	56	<code>\glsignore</code> .....	43
<code>\glsfieldfetch</code> .....	67	<code>\glsinlinedescformat</code> .....	229
<code>\glsfieldxdef</code> .....	131	<code>\glsinlinesubdescformat</code> .....	229
<code>\glsfindwidesttoplevelname</code> .....	231	<code>\glsinsert</code> .....	46, 49, 57–60, 154–166
<code>\GLSfirst</code> .....	210, 211	<code>\glskeylisttok</code> .....	90, 91, 150, 152
<code>\Glsfirst</code> .....	211	<code>\glslabel</code> .....	28, 44, 47, 48, 63, 64,
<code>\glsfirst</code> .....	210	.....	67, 68, 92, 146, 164–166, 176, 198, 200, 202

<code>\glslabeltok</code> ..	90, 150, 152, 169–174, 176, 177, 179, 180, 182, 190, 192, 194, 197–200, 202, 203
<code>\glsletentryfield</code>	139
<code>\glslink</code>	91
<code>\glslink options</code>	
<code>format</code>	137
<code>hyper</code>	204
<code>noindex</code>	6, 63, 204
<code>wrgloss</code>	47
<code>\glslinkcheckfirsthyperhook</code>	63
<code>\glslinkpostsetkeys</code>	48
<code>\glslinkvar</code>	66
<code>\glslistdottedwidth</code>	224
<code>\glslocalunset</code>	46
<code>\glslongaccessdisplay</code>	122
<code>\glslongdefaultfont</code>	154, 170, 172, 173, 177, 179, 180
<code>\glslongemfont</code>	189–195
<code>\glslongfont</code>	70, 71, 154, 159, 160, 163, 164, 170, 172, 174, 176, 177, 179, 180, 191–195, 197, 198, 201, 203
<code>\glslongfootnotefont</code>	173, 174, 176
<code>\glslongpltok</code> ...	152, 169, 171, 173, 174, 180, 181, 190, 192, 194, 197–200, 202, 203
<code>\glslongpluralaccessdisplay</code>	122
<code>\glslongtok</code>	90, 91, 150, 152, 169–173, 175, 177, 179–181, 190, 192, 194, 197–200, 202
<code>\glslonguserfont</code>	196–198, 200–203
<code>\glsnameaccessdisplay</code> ..	115, 116, 134–136
<code>\glsnamefont</code>	134–137
<code>\glsnavhyperlink</code>	101
<code>\glsnextpages</code>	97
<code>\glsnoidxdisplayloc</code>	99
<code>\glsnoidxdisplayloclisthandler</code>	98
<code>\glsnoidxloclist</code>	99, 114, 115
<code>\glsnoidxnumberlistloophandler</code>	99
<code>\glsnonextpages</code>	97
<code>\glsnonumberlistfalse</code>	42
<code>\glsnonumberlisttrue</code>	42
<code>\glsnumberlistloop</code>	95
<code>\glsnumlistlastsep</code>	98
<code>\glsnumlistsep</code>	98
<code>\glsopenbrace</code>	31, 106–108
<code>\glsorder</code>	93
<code>\glspagelistwidth</code>	224, 226–228
<code>\glspar</code>	113
<code>\GLSpl</code>	78, 89
<code>\Glspl</code>	39, 78, 89
<code>\glspl</code>	39, 78, 89
<code>\GLSplural</code>	209, 210
<code>\Glsplural</code>	210
<code>\glsplural</code>	209, 210
<code>\glspluralaccessdisplay</code>	117
<code>\glspluralsuffix</code>	110, 151, 154
<code>\glspostdescription</code>	145, 224–230
<code>\glspostinline</code>	229
<code>\glspostlinkhook</code>	45–47, 57–61, 72, 155–164
<code>\glsprestandardsort</code>	95
<code>\glsresetentrylist</code>	101, 112
<code>\glssee</code>	32–34
<code>\glsseeformat</code>	31, 94, 99
<code>\glsseelist</code>	31
<code>\glssetabbrvfmt</code>	44, 49, 69, 70, 132–137, 142, 143, 154–159, 161, 162
<code>\glssetattribute</code>	170–174, 176, 177, 179, 180, 182, 190, 192, 194, 197, 198, 200, 202, 203
<code>\glssetcategoryattribute</code>	78, 90, 92, 127, 128, 130, 131, 143
<code>\glssetnoexpandfield</code>	8, 9
<code>\glssettoctitle</code>	97
<code>\glsshortaccessdisplay</code>	121
<code>\glsshortpltok</code>	152, 169, 171–175, 177, 179, 190, 192, 197–200, 202, 203
<code>\glsshortpluralaccessdisplay</code> ..	121, 122
<code>\glsshorttok</code> ..	90, 91, 150–152, 169–173, 175, 177, 179, 181, 190, 192, 194, 197–202
<code>\glssubentryitem</code>	224–229, 239
<code>\glsymbolaccessdisplay</code>	118, 119
<code>\glsymbolpluralaccessdisplay</code>	119
<code>\glstarget</code>	224–229, 239
<code>\GLStext</code>	209
<code>\Glstext</code>	209
<code>\glstext</code>	209
<code>\glstextaccessdisplay</code>	116
<code>\glstextformat</code>	47, 48
<code>\glstextup</code>	182
<code>\glstreeindent</code>	238, 239
<code>\glstreenamebox</code>	239
<code>\glstreenamefmt</code>	230–239
<code>\GlstrLetField</code>	22
<code>\glstype</code>	46, 47, 57–61, 155–164
<code>\glsunset</code>	35, 46, 81, 82
<code>\glswrite</code>	31, 93
<code>\glswriteentry</code>	7
<code>\Glsxtr</code>	39
<code>\glsxtr</code>	39



\glsxtr@@do@wrglossary .....	7, 9, 10	\glsxtralttreeSymbolDescLocation ..	230, 239
\glsxtr@addloclistfield .....	10	\glsxtrassignfieldfont .....	49–56
\glsxtr@addunused .....	35	\glsxtrautoindex .....	138
\glsxtr@applyabbrvfmt .....	164	\glsxtrautoindexassignsort .....	139
\glsxtr@applyabbrvstyle ....	149, 151, 167	\glsxtrautoindexentry .....	139
\glsxtr@counterrecord .....	111	\glsxtrcat .....	38, 39
\glsxtr@doooption .....	4, 12, 16, 17	\glsxtrchecknohyperfirst .....	50–52
\glsxtr@fields .....	110	\glsxtrComputeTreeIndent .....	239
\glsxtr@headentry@p .....	73, 74	\glsxtrComputeTreeSubIndent .....	239
\glsxtr@ifnextpunc .....	148	\GlsXtrDefineAbbreviationShortcuts ..	14
\glsxtr@ifpunctoken .....	148	\GlsXtrDefineOtherShortcuts .....	14
\glsxtr@indexonly@saveentrycounter		\glsxtrdiscardperiod .....	146
.....	9, 10, 18	\glsxtrdisplayendloc .....	102
\glsxtr@keylist .....	38, 39	\glsxtrdisplayendlochook .....	103
\glsxtr@langtag .....	110	\glsxtrdisplaysingleloc .....	102, 103
\glsxtr@linkprefix .....	110	\glsxtrdisplaystartloc .....	102
\glsxtr@makeglossaries .....	93	\glsxtrdoautoindexname .....	65, 66, 137
\glsxtr@newabbreviation .....	91, 150	\glsxtrdopostpunc .....	176
\glsxtr@next .....	148	\glsxtrdowrglossaryhook .....	65
\glsxtr@org@getgrouptitle .....	100	\GlsXtrEnableEntryCounting .....	90
\glsxtr@org@newignoredglossary ....	25	\GlsXtrEnableEntryUnitCounting ....	78
\glsxtr@orgmakenoidxglossaries ..	35, 36	\GlsXtrEnableOnTheFly .....	37, 40
\glsxtr@pluralsuffixes .....	110	\glsxtrfieldlistgadd .....	111
\glsxtr@provideignoredglossary ....	27	\glsxtrfieldtitlecase .....	132–135
\glsxtr@punclist .....	148	\glsxtrfieldtitlecasesecs .....	132
\glsxtr@record .....	7	\glsxtrfieldxifinlist .....	113
\glsxtr@recordsee .....	8	\glsxtrfirstscfont .....	182–186
\glsxtr@resource .....	108, 110	\glsxtrfirstsmfont .....	186–189
\glsxtr@s@newignoredglossary .....	25	\GlsXtrFmtDefaultOptions .....	20
\glsxtr@s@provideignoredglossary ...	27	\GlsXtrFmtField .....	20, 21
\glsxtr@saveentrycounter .....	7, 8, 64	\GlsXtrFormatLocationList ..	42, 44, 236–238
\glsxtr@setup@record .....	9, 10, 17, 18	\GLSxtrfull .....	13, 213, 214
\glsxtr@shortcutsval .....	110	\Glsxtrfull .....	13, 214, 215
\glsxtr@texencoding .....	110	\glsxtrfull .....	13, 214
\glsxtr@usesee .....	30	\Glsxtrfullformat ..	153, 166–168, 170, 172,
\glsxtr@warnonexistsordo .....	6, 9, 10, 29	.....	174, 176, 178, 180, 181, 197, 199, 201, 203
\glsxtr@writefields .....	108	\glsxtrfullformat ..	153, 165–168, 170–172,
\glsxtrabbrvfootnote .....	173, 174, 176	.....	174, 176, 178, 179, 181, 197, 198, 201, 203
\glsxtrabbrvpluralsuffix ..	110, 154, 170,	\GLSxtrfullpl .....	13, 214, 215
.....	172, 174, 176, 177, 179, 180, 182, 186, 196	\Glsxtrfullpl .....	13, 215
\glsxtrabbrvtype .....	12, 152	\glsxtrfullpl .....	13, 214
\glsxtraddallcrossrefs .....	34	\Glsxtrfullplformat .....	.....
\glsxtralias .....	64	.....	153, 165, 167, 168, 170, 172,
\glsxtrAltTreeIndent .....	230	.....	174, 176, 178, 180, 181, 197, 199, 201, 203
\glsxtralttreeInit .....	238	\glsxtrfullplformat .....	.....
\glsxtrAltTreePar .....	229	.....	165, 167, 168, 170, 172,
\glsxtrAltTreeSetHangIndent ...	230, 239	.....	174, 176, 178, 179, 181, 197, 198, 201, 203
\glsxtrAltTreeSetSubHangIndent ....	239		
\glsxtralttreeSubSymbolDescLocation	239		



<code>\glxstrfullsep</code> .....	<code>\GLSxtrlong</code> .....	13, 212, 213
152, 153, 169–173, 175–181, 190, 192, 196	<code>\Glsxtrlong</code> .....	13, 213
<code>\glxstrgenabbrvfmt</code> .....	<code>\glxstrlong</code> .....	13, 212
44	<code>\GLSxtrlongpl</code> .....	13, 212, 213
<code>\glxstrgetgrouptitle</code> .....	<code>\Glsxtrlongpl</code> .....	13, 213
101	<code>\glxstrlongpl</code> .....	13, 212
<code>\Glsxtrheadfirst</code> .....	<code>\glxstrlongshortdescsort</code> .....	171
206	<code>\glxstrmarkhook</code> .....	204
<code>\glxtrheadfirst</code> .....	<code>\glxstrnewabbrevpresetkeyhook</code> .....	151
206	<code>\glxstrnewnumber</code> .....	13
<code>\Glsxtrheadfirstplural</code> .....	<code>\glxstrnewsymbol</code> .....	13
206	<code>\glxstrNoGlossaryWarning</code> .....	15, 104
<code>\glxtrheadfirstplural</code> .....	<code>\GlsXtrNoGlsWarningAutoMake</code> .....	108
206	<code>\GlsXtrNoGlsWarningBuildInfo</code> .....	108
<code>\Glsxtrheadfull</code> .....	<code>\GlsXtrNoGlsWarningCheckFile</code> .....	107
206	<code>\GlsXtrNoGlsWarningEmptyMain</code> ..	107, 108
<code>\glxtrheadfull</code> .....	<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ..	107
206	<code>\GlsXtrNoGlsWarningEmptyStart</code> .....	107
<code>\Glsxtrheadfullpl</code> .....	<code>\GlsXtrNoGlsWarningHead</code> .....	107
206	<code>\GlsXtrNoGlsWarningMisMatch</code> .....	108
<code>\glxtrheadfullpl</code> .....	<code>\GlsXtrNoGlsWarningNoOut</code> .....	108
206	<code>\GlsXtrNoGlsWarningTail</code> .....	108
<code>\Glsxtrheadlong</code> .....	<code>\glxstrorg@ifKV@glslink@hyper</code> .....	45
206	<code>\GLSxtrp</code> .....	73
<code>\glxtrheadlong</code> .....	<code>\Glsxtrp</code> .....	73
206	<code>\glxtrp</code> .....	72, 74, 75
<code>\Glsxtrheadlongpl</code> .....	<code>\Glsxtrpl</code> .....	39
206	<code>\glxtrpl</code> .....	39
<code>\glxtrheadlongpl</code> .....	<code>\glxtrpostdescription</code> .....	130, 145, 229
206	<code>\glxtrpostlink</code> .....	146
<code>\Glsxtrheadplural</code> .....	<code>\glxtrpostlinkendsentence</code> .....	146
206	<code>\glxtrpostlinkhook</code> .....	146
<code>\glxtrheadplural</code> .....	<code>\glxtrpostlocalreset</code> .....	77, 79, 87
206	<code>\glxtrpostlocalunset</code> .....	77, 79, 87
<code>\Glsxtrheadshort</code> .....	<code>\glxtrpostlongdescription</code> .....	25
206	<code>\glxtrpostnamehook</code> .....	135–137
<code>\glxtrheadshort</code> .....	<code>\GlsXtrPostNewAbbreviation</code> .....	152,
206	167, 168, 170, 171, 173–175, 177, 179,	
<code>\glxtrheadshort</code> .....	180, 182, 190, 192, 194, 197–200, 202, 203	
206	<code>\glxtrpostreset</code> .....	77, 79, 87
<code>\Glsxtrheadshortpl</code> .....	<code>\glxtrpostunset</code> .....	77, 79, 87
206	<code>\glxtrprotectlinks</code> .....	67, 68
<code>\glxtrheadshortpl</code> .....	<code>\GlsXtrRecordCounter</code> .....	8
206	<code>\glxtrregularfont</code> .....	44, 49
<code>\Glsxtrheadtext</code> .....	<code>\glxtrresourcecount</code> .....	109
206	<code>\glxtrresourcefile</code> .....	109
<code>\glxtrheadtext</code> .....	<code>\glxtrrestoremarkhook</code> .....	204, 205
206	<code>\glxtrscfont</code> .....	182–186
<code>\glxtrifcounttrigger</code> .....	<code>\glxtrscsuffix</code> .....	182–186
81, 82		
<code>\glxtrifemptyglossary</code> .....		
102, 107, 112		
<code>\glxtrifindexing</code> .....		
65		
<code>\glxtrifinmark</code> .....		
74–76, 205, 206		
<code>\glxtrifnextpunc</code> .....		
148, 149		
<code>\glxtrifperiod</code> .....		
147		
<code>\glxtrifwasfirstuse</code> .....		
49–52, 57–60, 63, 92, 146,		
147, 155, 158–163, 175, 176, 198–200, 202		
<code>\glxtrindexaliased</code> .....		
64		
<code>\glxtrindexseealso</code> .....		
33, 34		
<code>\glxtrinitwrgloss</code> .....		
47		
<code>\glxtrinitwrglossbeforefalse</code> .....		
47		
<code>\glxtrinitwrglossbeforetrue</code> .....		
47		
<code>\Glsxtrinlinefullformat</code> ....		
153, 155,		
167, 168, 175, 177–179, 181, 199, 201, 220		
<code>\glxtrinlinefullformat</code> 153, 154, 156,		
167, 168, 175–177, 179, 180, 199, 201, 220		
<code>\Glsxtrinlinefullplformat</code> ..		
154, 157,		
167, 168, 175, 177–179, 181, 199, 201, 221		
<code>\glxtrinlinefullplformat</code> ..		
153, 156,		
157, 167, 168, 175, 177–180, 199, 201, 220		
<code>\glxtrininsertinsidefalse</code> .....		
169		
<code>\glxtrlocationhyperlink</code> .....		
103		
<code>\glxtrlocrangefmt</code> .....		
102, 103		





<code>\newabbreviation</code> .....	13	<code>\nofiles</code> .....	107
<code>\newabbreviationhook</code> .....	152	<code>\noindent</code> .....	107, 108
<code>\newabbreviationstyle</code> .....		<code>\nopostdesc</code> .....	25, 38, 39, 97, 130
	169, 171–173, 175, 177, 179–195, 197–203	<code>\nr</code> .....	6, 9, 11, 14, 15, 47, 97
<code>\newacronym</code> .....	90, 91	<code>\ns@GLSxtrfull</code> .....	155
<code>\newacronymhook</code> .....	90	<code>\ns@Glsxtrfull</code> .....	155
<code>\newacronymstyle</code> .....	91, 92	<code>\ns@glxtrfull</code> .....	154
<code>\newcommand</code> .....	4–8,	<code>\ns@GLSxtrfullpl</code> .....	157
	10–31, 33–35, 37–40, 42–44, 47, 49, 50,	<code>\ns@Glsxtrfullpl</code> .....	156
	61, 62, 64–66, 68, 72–80, 83–93, 96, 98–	<code>\ns@glxtrfullpl</code> .....	156
	100, 102, 103, 105–109, 111–132, 137–	<code>\ns@GLSxtrlong</code> .....	160
	150, 152–164, 166–170, 173, 182, 186,	<code>\ns@Glsxtrlong</code> .....	159
	189, 190, 196, 205–221, 223, 229–231, 238	<code>\ns@glxtrlong</code> .....	159
<code>\newcount</code> .....	10, 109	<code>\ns@GLSxtrlongpl</code> .....	163
<code>\newentry</code> .....	13	<code>\ns@Glsxtrlongpl</code> .....	163
<code>\newglossary</code> .....	12, 93	<code>\ns@glxtrlongpl</code> .....	162
<code>\newglossaryentry</code> .....		<code>\ns@GLSxtrshort</code> .....	158
	13, 36, 78, 79, 86, 90, 130, 131, 152	<code>\ns@Glsxtrshort</code> .....	158
<code>\newglossaryentry options</code>		<code>\ns@glxtrshort</code> .....	157
<code>alias</code> .....	11, 30, 32–35	<code>\ns@GLSxtrshortpl</code> .....	162
<code>desc</code> .....	119, 120, 124, 125	<code>\ns@Glsxtrshortpl</code> .....	161
<code>descplural</code> .....	120, 125	<code>\ns@glxtrshortpl</code> .....	161
<code>first</code> .....	66, 117, 123, 124, 169, 210–212, 217, 241	<code>\null</code> .....	15
<code>firstplural</code> .....	118, 124, 169, 211, 218, 241	<code>\number</code> .....	85–88, 108
<code>loclist</code> .....	21	<code>\numexpr</code> .....	85, 86, 88
<code>long</code> .....	122, 125, 219		
<code>longplural</code> .....	122, 126, 219		
<code>name</code> .....	115, 116, 123, 138		
<code>plural</code> .....	116, 117, 123, 169, 209, 210, 217		
<code>see</code> .....	11, 17, 30, 32, 35, 36, 94		
<code>seealso</code> .....	11, 30–32, 34, 35, 252		
<code>short</code> .....	121, 125, 150		
<code>shortplural</code> .....	122, 125, 150		
<code>symbol</code> .....	118, 119, 124		
<code>symbolplural</code> .....	119, 124		
<code>text</code> ....	66, 116, 123, 169, 171, 208, 209, 216		
<code>\newif</code> .....	47, 137, 169		
<code>\newlength</code> .....	230		
<code>\newnum</code> .....	13		
<code>\newrobustcmd</code> .....	20–		
	23, 31, 32, 61, 62, 72, 73, 83, 100, 114,		
	143, 144, 154–163, 205, 207–215, 231–237		
<code>\newsym</code> .....	13		
<code>\newterm</code> .....	130		
<code>\newtoks</code> .....	150		
<code>\newwrite</code> .....	93		
<code>\NoCaseChange</code> .....	74–76, 207–215		
<code>\noexpand</code> .....	8, 16, 31, 33, 34,		
	90, 91, 104, 105, 108, 112, 139, 140, 152, 223		

## O

<code>\or</code> .....	6, 9, 10, 14, 36, 47
<code>\org@glossaryentrynumbers</code> .....	41, 42, 97
<code>\org@glossarytitle</code> .....	96, 97
<code>\org@ifKV@glslink@hyper</code> .....	47, 48

## P

<code>\p@gl@s@hyp@opt</code> .....	66
package options:	
<code>abbreviations</code> .....	12
<code>accsupp</code> .....	15, 115
<code>acronym</code> .....	12
<code>automake</code> .....	96, 106, 110
<code>true</code> .....	110
<code>autoseeindex</code> .....	17
<code>false</code> .....	17
<code>docdef</code> .....	10, 35, 36, 78, 86
<code>false</code> .....	36
<code>restricted</code> .....	11
<code>true</code> .....	36
<code>nonumberlist</code> .....	41
<code>nopostdot</code>	
<code>false</code> .....	12
<code>numbers</code> .....	13



<code>\space</code> .....	5, 8, 31, 32, 37, 40, 64, 78–80, 86, 88–90, 92–95, 97, 105, 107, 111, 113, 146, 153, 170, 229, 230, 238
<code>\spacefactor</code> .....	146, 151, 229
<code>\string</code> .....	5, 7, 8, 31, 32, 36, 37, 40, 43, 61, 62, 64, 72, 73, 78–80, 86, 88–90, 92– 97, 104–108, 110, 111, 113, 134–137, 139
<code>\strut</code> .....	224–229
<code>\subglossentry</code> .....	97, 114, 224–229, 239
<b>T</b>	
<code>\tablehead</code> .....	227, 228
<code>\tabletail</code> .....	227, 228
<code>\tabularnewline</code> .....	224–229
<code>\TeX</code> .....	106
<code>\texorpdfstring</code> .....	20, 73–76, 215–221
<code>textcase</code> package .....	204
<code>\textsc</code> .....	182
<code>\textsmaller</code> .....	186
<code>\texttt</code> .....	105–108
<code>\the</code> .....	90, 91, 103, 109, 142, 152, 169–177, 179–182, 190, 192, 194, 197–203
<code>\theglsentrycounter</code> .....	7, 49
<code>\theHglentrycounter</code> .....	7, 49
<code>\theindex</code> .....	138
<code>\this@dialect</code> .....	221
<code>\toks@</code> .....	103, 142
<code>tracklang</code> package .....	110
<b>U</b>	
<code>\undef</code> .....	9, 10, 143
<code>\underline</code> .....	144
<code>\unskip</code> .....	25, 35, 224
<code>\usepackage</code> .....	106–108
<b>V</b>	
<code>\val</code> .....	6, 9, 11, 14, 15, 47, 97
<b>W</b>	
<code>\warn@nomakeglossaries</code> .....	94
<code>\warn@noprintglossary</code> .....	94, 97
<code>\write</code> .....	31, 80, 88, 89, 93, 104, 105, 110
<b>X</b>	
<code>\x</code> .....	103
<code>\xcapitalisewords</code> .....	132
<code>\xdef</code> .....	97, 113
<code>\xifinlist</code> .....	84
<code>\xifinlistcs</code> .....	22
<code>xindy</code> .....	241
<code>xindy</code> .....	93
<code>xkeyval</code> package .....	4
<code>\XKV@checkchoice</code> .....	44
<code>\XKV@plfalse</code> .....	44
<code>\XKV@resa</code> .....	44
<code>\XKV@sttrue</code> .....	44