

glossaries-extra.sty v1.14: documented code

Nicola L.C. Talbot

Dickimaw Books

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

2017-04-18

Abstract

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

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

Contents

1 Main Package Code (glossaries-extra.sty)	4
1.1 Package Initialisation and Options	4
1.2 Extra Utilities	17
1.3 Modifications to Commands Provided by glossaries	23
1.3.1 Existence Checks	28
1.3.2 Document Definitions	31
1.3.3 Existing Glossary Style Modifications	35
1.3.4 Entry Formatting, Hyperlinks and Indexing	39
1.3.5 Entry Counting	72
1.3.6 Acronym Modifications	85
1.3.7 Indexing and Displaying Glossaries	88
1.4 Integration with glossaries-accsupp	110
1.5 Categories	121
1.6 Abbreviations	143
1.6.1 Abbreviation Styles Setup	160
1.6.2 Predefined Styles (Default Font)	163
1.6.3 Predefined Styles (Small Capitals)	176
1.6.4 Predefined Styles (Fake Small Capitals)	180
1.6.5 Predefined Styles (Emphasized)	184
1.6.6 Predefined Styles (User Parentheses Hook)	190
1.7 Using Entries in Headings	198
1.8 Multi-Lingual Support	215
2 Style Adjustments (glossaries-extra-stylemods.sty)	217
2.1 Package Initialisation	217
2.2 List-Like Styles	218
2.3 Longtable Styles	218
2.4 Long Ragged Styles	219
2.5 Supertabular Styles	221
2.6 Super Ragged Styles	222
2.7 Inline Style	223
2.8 Tree Styles	223
Glossary	235
Change History	236
Index	247

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries-extra}[2017/04/18 v1.14 (NLCT)]

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

Requires etoolbox package.
4 \RequirePackage{etoolbox}

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

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

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

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

Declare package options.

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

```

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

```

`\f@for@gl@sentries`

```

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

```

`\f@for@gl@sentries`

```

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

```

```

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

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

```

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

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

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

`\glstr@recordsee` Does nothing by default.

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

`@@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 {%

```

Save the entry counter.

```

95 \ifdefempty{\@glxtr@thevalue}%
96 {%
97 \glxtr@saveentrycounter
98 }%
99 {%
100 \let\theglentrycounter\@glxtr@thevalue
101 \def\theHglentrycounter{\@glxtr@theHvalue}%
102 }%

```

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

```

103 \let\@@do@@wrglossary\@glxtr@dorecord
104 \glxtr@@do@@wrglossary{#2}%
105 }%
106 \fi
107 \endgroup
108 }

```

glxtr@dorecord

```

109 \newcommand*\@glxtr@dorecord{%
110 \protected@write\@auxout{}\string\glxtr@record
111 {\@gls@label}\@glo@counterprefix{\@gls@counter}\@glsnumberformat}%
112 {\@glslocref}}%
113 \@glxtr@counterrecordhook
114 }

```

r@recordcounter

```

115 \newcommand*\@glxtr@recordcounter{%
116 \@glxtr@noop@recordcounter
117 }

```

p@recordcounter

```

118 \newcommand*\@glxtr@noop@recordcounter}[1]{%
119 \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
120 requires record=only or record=alsoindex package option}{}%
121 }

```

p@recordcounter

```

122 \newcommand*\@glxtr@op@recordcounter}[1]{%
123 \eappto\@glxtr@counterrecordhook{\noexpand\@glxtr@docounterrecord{#1}}%
124 }

```

lsxtr@recordsee Deal with \glssee in record mode.

```

125 \newcommand*\@glxtr@recordsee}[2]{%

```

```

126 \def\@gls@xref{#2}%
127 \@onelevel@sanitize\@gls@xref
128 \protected@write\@auxout{}\string\glsxtr@recordsee{#1}{\@gls@xref}}%
129 }

```

srtglossaryunit

```

130 \newcommand{\printunsrtglossaryunit}{%
131   \print@noop@unsrtglossaryunit
132 }

```

tr@setup@record Initialise.

```

133 \newcommand*{\glsxtr@setup@record}{}

```

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

```

134 \newcommand*{\glsxtr@indexonly@saveentrycounter}{%
135   \ifKV@glslink@noindex
136   \else
137     \glsxtr@saveentrycounter
138   \fi
139 }

```

addloclistfield

```

140 \newcommand*{\glsxtr@addloclistfield}{%
141   \key@ifundefined{glossentry}{loclist}%
142   {%
143     \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
144     \appto\@gls@keymap{,{loclist}{loclist}}%
145     \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
146     \appto\@newglossaryentryposthook{%
147       \gls@assign@field{\@glo@label}{loclist}{\@glo@loclist}%
148     }%
149     \glssetnoexpandfield{loclist}%
150   }%
151   {%

```

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

```

152   \key@ifundefined{glossentry}{location}%
153   {%
154     \define@key{glossentry}{location}{\def\@glo@location{##1}}%
155     \appto\@gls@keymap{,{location}{location}}%
156     \appto\@newglossaryentryprehook{\def\@glo@location{}}%
157     \appto\@newglossaryentryposthook{%
158       \gls@assign@field{\@glo@label}{location}{\@glo@location}%
159     }%
160     \glssetnoexpandfield{location}%
161   }%
162   {%

```

Add a key to store the group heading.


```

163 \key@ifundefined{glossentry}{group}%
164 {%
165   \define@key{glossentry}{group}{\def\@glo@group{##1}}%
166   \appto\@gls@keymap{,{group}{group}}%
167   \appto\@newglossaryentryprehook{\def\@glo@group{}}%
168   \appto\@newglossaryentryposthook{%
169     \gls@assign@field{\@glo@label}{group}{\@glo@group}%
170   }%
171   \glssetnoexpandfield{group}%
172 }%
173 {}%
174 }

```

Now define the record package option.

```

175 \define@choicekey{glossaries-extra.sty}{record}[\val\nr]%
176 {off,only,alsoindex}%
177 [only]%
178 {%
179   \ifcase\nr\relax

```

Don't record.

```

180   \def\glstr@setup@record{%
181     \let\@glo@autosee\@glstr@org@gloautosee
182     \renewcommand*{\@do@seeglossary}{\@glstr@org@doseeglossary}%
183     \renewcommand*{\@glstr@record}[3]{}%
184     \let\@do@wrglossary\glstr@@do@wrglossary
185     \let\@gls@saveentrycounter\glstr@indexonly@saveentrycounter
186     \let\glstrundefaction\@glstr@err@undefaction
187     \let\glstr@warnonexistsordo\@gobble
188     \let\@glstr@recordcounter\@glstr@noop@recordcounter
189     \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
190     \undef\glstrsetaliasnoindex
191   }%
192   \or

```

Only record (don't index).

```

193   \def\glstr@setup@record{%
194     \ifdef\@glo@autosee{\let\@glo@autosee\relax}{}%
195     \let\@do@seeglossary\@glstr@recordsee
196     \let\@glstr@record\@glstr@record
197     \let\@do@wrglossary\@gobble
198     \let\@gls@saveentrycounter\relax
199     \let\glstrundefaction\@glstr@warn@undefaction
200     \let\glstr@warnonexistsordo\@glstr@warn@onexistsordo
201     \glstr@addloclistfield
202     \renewcommand*{\@glstr@autoindexcrossrefs}{}%
203     \let\@glstr@recordcounter\@glstr@op@recordcounter
204     \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%

```

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

```

205     \def\glxtrsetaliasnoindex{}%
206   }%
207   \or
Record and index.
208   \def\glxtr@setup@record{%
209     \let\@glo@autosee\@glxtr@org@gloautosee
210     \renewcommand*{\@do@seeglossary}{\@glxtr@org@doseeglossary}%
211     \let\@glxtr@record\@glxtr@record
212     \let\@do@wrglossary\glxtr@do@wrglossary
213     \let\@gls@saveentrycounter\glxtr@indexonly@saveentrycounter
214     \let\glxtrundefaction\@glxtr@warn@undefaction
215     \let\glxtr@warnonexistsordo\@glxtr@warn@onexistsordo
216     \glxtr@addloclistfield
217     \let\@glxtr@recordcounter\@glxtr@op@recordcounter
218     \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
219     \undef\glxtrsetaliasnoindex
220   }%
221   \fi
222 }

```

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

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

```

223 \newcount\@glxtr@docdefval

```

Need to provide conditional commands that are backward compatible:

```

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

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

sxtrdocdeffalse
226 \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.

```

227 \define@choicekey{glossaries-extra.sty}{docdef}[\val\nr]%
228 {false,true,restricted}[true]%
229 {%
230   \@glxtr@docdefval=\nr\relax
231   \ifnum\@glxtr@docdefval=2\relax
232     \renewcommand*{\@glsdoifexistsorwarn}{\glsdoifexists}%
233   \fi
234 }

```

docdefrestricted

```
235 \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).

```
236 \newcommand*{\@glxdoifexistsorwarn}{\glxdoifexistsorwarn}
```

indexcrossrefs Automatically index cross references at the end of the document

```
237 \define@boolkey{glossaries-extra.sty}{@glxtr}{indexcrossrefs}[true]{%
238   \if@glxtrindexcrossrefs
239   \else
240   \renewcommand*{\@glxtr@autoindexcrossrefs}{}%
241   \fi
242 }
```

Switch off since this can increase the build time.

```
243 \@glxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

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

iesExtraWarning Allow users to suppress warnings.

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

raWarningNoLine Allow users to suppress warnings.

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

248 \@glxtr@declareoption{nowarn}{%
249   \let\GlossariesExtraWarning\@gobble
250   \let\GlossariesExtraWarningNoLine\@gobble
251   \glxtr@doption{nowarn}%
252 }
```

postdot Shortcut for nopostdot=false

```
253 \@glxtr@declareoption{postdot}{%
254   \glxtr@doption{nopostdot=false}%
255 }
```

glxtrabbrvtype Glossary type for abbreviations.

```
256 \newcommand*{\glxtrabbrvtype}{\glxdefaulttype}
```

bbreviationsdef Set by abbreviations option.

```
257 \newcommand*{\@glxtr@abbreviationsdef}{}%
```

bbreviationsdef

```

258 \newcommand*{\@glsxtr@doabbreviationsdef}{%
259   \ifpackageloaded{babel}%
260     {\providecommand{\abbreviationsname}{\acronymname}}%
261     {\providecommand{\abbreviationsname}{Abbreviations}}%
262   \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
263   \renewcommand*{\glsxtrabbrvtype}{abbreviations}%
264   \newcommand*{\printabbreviations}[1][1]{%
265     \printglossary[type=\glsxtrabbrvtype,##1]%
266   }%
267   \disable@keys{glossaries-extra.sty}{abbreviations}%

   If the acronym option hasn't been used, change \acronymtype to \glsxtrabbrvtype.
268   \ifglsacronym
269   \else
270     \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
271   \fi
272 }%
```

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

```

273 \@glsxtr@declareoption{abbreviations}{%
274   \let\@glsxtr@abbreviationsdef\@glsxtr@doabbreviationsdef
275 }
```

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

```

276 \newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
277   \newcommand*{\ab}{\cgl{s}}%
278   \newcommand*{\abp}{\cgl{s}pl}%
279   \newcommand*{\as}{\glsxtrshort}%
280   \newcommand*{\asp}{\glsxtrshortpl}%
281   \newcommand*{\al}{\glsxtrlong}%
282   \newcommand*{\alp}{\glsxtrlongpl}%
283   \newcommand*{\af}{\glsxtrfull}%
284   \newcommand*{\afp}{\glsxtrfullpl}%
285   \newcommand*{\Ab}{\cGls}%
286   \newcommand*{\Abp}{\cGlspl}%
287   \newcommand*{\As}{\Glsxtrshort}%
288   \newcommand*{\Asp}{\Glsxtrshortpl}%
289   \newcommand*{\Al}{\Glsxtrlong}%
290   \newcommand*{\Alp}{\Glsxtrlongpl}%
291   \newcommand*{\Af}{\Glsxtrfull}%
292   \newcommand*{\Afp}{\Glsxtrfullpl}%
293   \newcommand*{\AB}{\cGLS}%
294   \newcommand*{\ABP}{\cGLSpl}%
295   \newcommand*{\AS}{\GLSxtrshort}%
296   \newcommand*{\ASP}{\GLSxtrshortpl}%
297   \newcommand*{\AL}{\GLSxtrlong}%
298   \newcommand*{\ALP}{\GLSxtrlongpl}%

```

```

299 \newcommand*{\AF}{\GLSxtrfull}%
300 \newcommand*{\AFP}{\GLSxtrfullpl}%
301 \newcommand*{\newabbr}{\newabbreviation}%

```

Disable this command after it's been used.

```

302 \let\GlsXtrDefineAbbreviationShortcuts\relax
303 }

```

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

```

304 \newcommand*{\GlsXtrDefineOtherShortcuts}{%
305 \newcommand*{\newentry}{\newglossaryentry}%
306 \ifdef\printsymbols
307 {%
308 \newcommand*{\newsym}{\glxtrnewsymbol}%
309 }{}%
310 \ifdef\printnumbers
311 {%
312 \newcommand*{\newnum}{\glxtrnewnumber}%
313 }{}%
314 \let\GlsXtrDefineOtherShortcuts\relax
315 }

```

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.

```

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

```

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

```

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

```

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

```

318 \define@choicekey{glossaries-extra.sty}{shortcuts}[\val\nr]{%
319 {acronyms,acro,abbreviations,abbr,other,all,true,false}[true]{%
320 \let\@glxtr@shortcutsval\val
321 \ifcase\nr\relax % acronyms
322 \renewcommand*{\@glxtr@setupshortcuts}{%
323 \glacrshortcutstrue
324 \DefineAcronymSynonyms
325 }%
326 \or % acro
327 \renewcommand*{\@glxtr@setupshortcuts}{%
328 \glacrshortcutstrue
329 \DefineAcronymSynonyms
330 }%

```

```

331 \or % abbreviations
332   \renewcommand*{\@glxtr@setupshortcuts}{%
333     \GlsXtrDefineAbbreviationShortcuts
334   }%
335 \or % abbr
336   \renewcommand*{\@glxtr@setupshortcuts}{%
337     \GlsXtrDefineAbbreviationShortcuts
338   }%
339 \or % other
340   \renewcommand*{\@glxtr@setupshortcuts}{%
341     \GlsXtrDefineOtherShortcuts
342   }%
343 \or % all
344   \renewcommand*{\@glxtr@setupshortcuts}{%
345     \glsacrshortcutstrue
346     \DefineAcronymSynonyms
347     \GlsXtrDefineAbbreviationShortcuts
348     \GlsXtrDefineOtherShortcuts
349   }%
350 \or % true
351   \renewcommand*{\@glxtr@setupshortcuts}{%
352     \glsacrshortcutstrue
353     \DefineAcronymSynonyms
354     \GlsXtrDefineAbbreviationShortcuts
355     \GlsXtrDefineOtherShortcuts
356   }%
357 \else % none, false
358   \renewcommand*{\@glxtr@setupshortcuts}{}%
359 \fi
360 }

```

`\lsxtr@doaccsupp`

```

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

```

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

```

362 \@glxtr@declareoption{accsupp}{%
363   \renewcommand*{\@glxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}

```

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

```

364 \newcommand{\glxtrNoGlossaryWarning}[1]{%
365   \@glxtr@defaultnoglossarywarning{#1}%
366 }

```

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

```

367 \define@choicekey{glossaries-extra.sty}{nomissingglstext}[\val\nr]%
368 {true,false}[true]{%
369   \ifcase\nr\relax % true
370     \renewcommand{\glxtrNoGlossaryWarning}[1]{%

```

```

371     \null
372   }%
373 \else % false
374   \renewcommand{\glxstrNoGlossaryWarning}[1]{%
375     \glxstr@defaultnoglossarywarning{#1}%
376   }%
377 \fi
378 }

```

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

`glxstr@redefstyles`

```

379 \newcommand*{\@glxstr@redefstyles}{}

```

`stylemods`

```

380 \define@key{glossaries-extra.sty}{stylemods}{%
381   \ifblank{#1}%
382   {%
383     \renewcommand*{\@glxstr@redefstyles}{%
384       \RequirePackage{glossaries-extra-stylemods}}%
385   }%
386   {%
387     \renewcommand*{\@glxstr@redefstyles}{}%
388     \@for\@glxstr@tmp:=#1\do{%
389       \IfFileExists{glossary-\@glxstr@tmp.sty}%
390       {%
391         \eappto\@glxstr@redefstyles{%
392           \noexpand\RequirePackage{glossary-\@glxstr@tmp}}%
393       }%
394       {%
395         \PackageError{glossaries-extra}%
396           {Glossaries style package ‘glossary-\@glxstr@tmp.sty’
397             doesn’t exist (did you mean to use the ‘style’ key?)}%
398           {The list of values (#1) in the ‘stylemods’ key should
399             match the glossary-xxx.sty files provided with
400             glossaries.sty}%
401       }%
402     }%
403     \appto\@glxstr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
404   }%
405 }

```

`glxstr@do@style`

```

406 \newcommand*{\@glxstr@do@style}{}

```

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

```

407 \define@key{glossaries-extra.sty}{style}{%
408   \renewcommand*{\@glxstr@do@style}{%

```

Set this as the default style:

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

Set this style:

```
410 \setglossarystyle{#1}%  
411 }%  
412 }
```

Pass all other options to glossaries.

```
413 \DeclareOptionX*{%  
414 \expandafter\glxtr@doption\expandafter{\CurrentOption}}
```

Process options.

```
415 \ProcessOptionsX
```

Load glossaries if not already loaded.

```
416 \RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
417 \@glxtr@doaccsupp
```

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

```
418 \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.)

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

Define abbreviations glossaries if required.

```
420 \@glxtr@abbreviationsdef  
421 \let\@glxtr@abbreviationsdef\relax
```

Setup shortcuts if required.

```
422 \@glxtr@setupshortcuts
```

Redefine \@glxtr@redef@forl@sentries if required.

```
423 \@glxtr@redef@forl@sentries
```

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

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

Now define the user command:

```
425 \newcommand*{\glossariesextrasetup}[1]{%  
426 \let\glxtr@setup@record\relax  
427 \let\@glxtr@setupshortcuts\relax  
428 \let\@glxtr@redef@forl@sentries\relax  
429 \setkeys{glossaries-extra.sty}{#1}%  
430 \@glxtr@abbreviationsdef  
431 \let\@glxtr@abbreviationsdef\relax
```



```

432 \@glxtr@setupshortcuts
433 \glxtr@setup@record
434 \@glxtr@redef@forglsentries
435 }

```

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

```

436 \let\glxtr@@do@wrglossary\@do@wrglossary

```

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

```

437 \let\glxtr@saveentrycounter\@gls@saveentrycounter

```

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

```

438 \let\@gls@saveentrycounter\glxtr@indexonly@saveentrycounter

```

Set up record option if required.

```

439 \glxtr@setup@record

```

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

```

440 \AtBeginDocument{%
441   \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
442   \def\@glxtrundeftag{\glxtrundeftag}%
443 }

```

1.2 Extra Utilities

`trifemptyglossary` `\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.)

```

444 \newcommand{\glxtrifemptyglossary}[3]{%
445   \ifcsdef{glolist@#1}%
446   {%
447     \ifcsstring{glolist@#1}{,}{#2}{#3}%
448   }%
449   {%
450     \glxtrundefaction{Glossary type '#1' doesn't exist}{}%
451     #2%
452   }%
453 }

```

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

```
454 \newcommand*{\glxtrifkeydefined}[3]{%
455   \key@ifundefined{glossentry}{#1}{#3}{#2}%
456 }
```

providestoragekey Like \gl saddstoragekey but does nothing if the key has already been defined.

```
457 \newcommand*{\glxtrprovidestoragekey}{%
458   \@ifstar\sglsxtr@provide@storagekey\glxtr@provide@storagekey
459 }
```

vide@storagekey Unstarred version.

```
460 \newcommand*{\@glxtr@provide@storagekey}[3]{%
461   \key@ifundefined{glossentry}{#1}%
462   {%
463     \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
464     \appto\@gls@keymap{,{#1}{#1}}%
465     \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
466     \appto\@newglossaryentryposthook{%
467       \letcs{\@glo@tmp}{@glo@#1}%
468       \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
469     }%

```

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

```
470   \ifblank{#3}
471   {}%
472   {%
473     \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
474   }%
475 }%
476 {%

```

Provide the no-link command if not already defined.

```
477   \ifblank{#3}
478   {}%
479   {%
480     \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
481   }%
482 }%
483 }
```

vide@storagekey Starred version.

```
484 \newcommand*{\sglsxtr@provide@storagekey}[1]{%
485   \key@ifundefined{glossentry}{#1}%
486   {%
487     \expandafter\newcommand\expandafter*\expandafter
488     {\csname gls@assign@#1@field\endcsname}[2]{%
489       \@gls@expand@field{##1}{#1}{##2}%
490     }%

```

```

491 }%
492 {}%
493 \@glstr@provide@addstoragekey{#1}%
494 }

```

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

```

495 \newcommand{\GlsXtrFmtField}{useri}

```

`\GlsXtrFmtDefaultOptions`

```

496 \newcommand{\GlsXtrFmtDefaultOptions}{noindex}

```

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

```

497 \newrobustcmd*{\glstrfmt}[3][[]]{%
498   \glsdoifexistsordo{#2}%
499   {%
500     \ifglshasfield{\GlsXtrFmtField}{#2}%
501     {%
502       \let\do@glstr@link@checkfirsthyper\relax
503       \expandafter\@glstr@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
504       {\csuse{\glscurrentfieldvalue}{#3}}%
505     }%
506     {#3}%
507   }%
508   {#3}%
509 }

```

`\glstrententryfmt` No link or indexing.

```

510 \ifdef\texorpdfstring
511 {
512   \newcommand*{\glstrententryfmt}[2]{%
513     \texorpdfstring{\@glstrententryfmt{#1}{#2}}{#2}%
514   }
515 }
516 {
517   \newcommand*{\glstrententryfmt}{\@glstrententryfmt}
518 }

```

`@glstrententryfmt`

```

519 \newrobustcmd*{\@glstrententryfmt}[2]{%
520   \glsdoifexistsordo
521   {%
522     \ifglshasfield{\GlsXtrFmtField}{#1}%
523     {%

```

```

524     \csuse{\glscurrentfieldvalue}{#2}%
525 }%
526 {#2}%
527 }%
528 {#2}%
529 }

```

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.

```

530 \newcommand*\glsxtrfieldlistadd}[3]{%
531   \listcsadd{glo@\glsdetoklabel{#1}@#2}{#3}%
532 }

```

trfieldlistgadd Similarly but uses `\listcsgadd`.

```

533 \newcommand*\glsxtrfieldlistgadd}[3]{%
534   \listcsgadd{glo@\glsdetoklabel{#1}@#2}{#3}%
535 }

```

trfieldlistead Similarly but uses `\listcseadd`.

```

536 \newcommand*\glsxtrfieldlistead}[3]{%
537   \listcseadd{glo@\glsdetoklabel{#1}@#2}{#3}%
538 }

```

trfieldlistxadd Similarly but uses `\listcsxadd`.

```

539 \newcommand*\glsxtrfieldlistxadd}[3]{%
540   \listcsxadd{glo@\glsdetoklabel{#1}@#2}{#3}%
541 }

```

Now provide commands to iterate over these lists.

fieldddolistloop

```

542 \newcommand*\glsxtrfieldddolistloop}[2]{%
543   \dolistcsloop{glo@\glsdetoklabel{#1}@#2}%
544 }

```

fieldforlistloop

```

545 \newcommand*\glsxtrfieldforlistloop}[3]{%
546   \forlistcsloop{glo@\glsdetoklabel{#1}@#2}{#3}%
547 }

```

List element tests:

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

```

548 \newcommand*\glsxtrfieldifinlist}[5]{%
549   \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
550 }

```

`rfieldxifinlist` Expands item.

```

551 \newcommand*{\glstrfieldxifinlist}[5]{%
552   \xifinlistcs{#3}{glo\glsetoklabel{#1}@#2}{#4}{#5}%
553 }

```

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

```

554 \newcommand*{\glstrusefield}[2]{%
555   \@gls@entry@field{#1}{#2}%
556 }

```

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

```

557 \newcommand*{\Glsxtrusefield}[2]{%
558   \@gls@entry@field{#1}{#2}%
559 }

```

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

```

560 \newcommand*{\glstrdeffield}[2]{\csdef{glo\glsetoklabel{#1}@#2}}

```

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

```

561 \newcommand*{\glstxredefield}[2]{\csedef{glo\glsetoklabel{#1}@#2}}

```

`etfieldifexists`

```

562 \newcommand*{\glstrsetfieldifexists}[3]{\glsoifexists{#1}{#3}}

```

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

```

563 \newrobustcmd*{\GlsXtrSetField}[3]{%
564   \glstrsetfieldifexists{#1}{#2}%
565   {\csdef{glo\glsetoklabel{#1}@#2}{#3}}%
566 }

```

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

```

567 \newrobustcmd*{\GlsXtrLetField}[3]{%
568   \glstrsetfieldifexists{#1}{#2}%
569   {\cslet{glo\glsetoklabel{#1}@#2}{#3}}%
570 }

```

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

```

571 \newrobustcmd*{\csGlsXtrLetField}[3]{%
572   \glstrsetfieldifexists{#1}{#2}%
573   {\csletcs{glo\glsetoklabel{#1}@#2}{#3}}%
574 }

```

`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.

```

575 \newrobustcmd*{\GlsXtrLetFieldToField}[4]{%

```

```

576 \glsxtrsetfieldifexists{#1}{#2}%
577 {\csletcs{glo@glstetoklabel{#1}@#2}{glo@glstetoklabel{#3}@#4}}%
578 }

```

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

```

579 \newrobustcmd*{\gGlsXtrSetField}[3]{%
580 \glsxtrsetfieldifexists{#1}{#2}%
581 {\csgdef{glo@glstetoklabel{#1}@#2}{#3}}%
582 }

```

xGlsXtrSetField

```

583 \newrobustcmd*{\xGlsXtrSetField}[3]{%
584 \glsxtrsetfieldifexists{#1}{#2}%
585 {\protected@csxdef{glo@glstetoklabel{#1}@#2}{#3}}%
586 }

```

eGlsXtrSetField

```

587 \newrobustcmd*{\eGlsXtrSetField}[3]{%
588 \glsxtrsetfieldifexists{#1}{#2}%
589 {\protected@csedef{glo@glstetoklabel{#1}@#2}{#3}}%
590 }

```

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

```

591 \ifglstentrycounter
592 \newcommand*{\glsxtrpageref}[1]{\pageref{glstentry-\glstetoklabel{#1}}}
593 \else
594 \ifglssubentrycounter
595 \newcommand*{\glsxtrpageref}[1]{\pageref{glstentry-\glstetoklabel{#1}}}
596 \else
597 \newcommand*{\glsxtrpageref}[1]{\gls{#1}}
598 \fi
599 \fi

```

lossarypreamble

```

600 \newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
601 \ifcsdef{glolist@#1}%
602 {%
603 \ifcsundef{@glossarypreamble@#1}%
604 {\csdef{@glossarypreamble@#1}{}}%
605 {}%
606 \csappto{@glossarypreamble@#1}{#2}%
607 }%
608 {%
609 \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
610 }%
611 }

```

lossarypreamble

```
612 \newcommand{\preglossarypreamble}[2][\glsdefaultttype]{%
613   \ifcsdef{glolist@#1}%
614   {%
615     \ifcsundef{@glossarypreamble@#1}%
616     {\csdef{@glossarypreamble@#1}{}}%
617   }%
618   \cspretoto{@glossarypreamble@#1}{#2}%
619 }%
620 {%
621   \GlossariesExtraWarning{Glossary ‘#1’ is not defined}%
622 }%
623 }
```

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.

`\glsextralias` Provide a key to allow aliases to be defined. The key should be set to the label of the synonymous entry.

```
624 \glsaddstoragekey*{alias}{\glsextralias}
```

`ryentryposthook` Append to the hook to check for the alias key.

```
625 \appto\@newglossaryentryposthook{%
626   \ifcsvoid{glo@\@glo@label @alias}{}%
627   {%
```

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

```
628   \ifdefvoid\@glo@see
629   {%
630     \edef\@do@glsee{\noexpand\glsee
631       {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
632     \@do@glsee
633   }%
634   {}%
635 }%
636 }
```

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

ewglossaryentry

```
637 \renewcommand*{\longnewglossaryentry}{%
638   \@ifstar{\glxtr@s@longnewglossaryentry\@glxtr@longnewglossaryentry
639 }
```

ewglossaryentry Starred version.

```
640 \newcommand{\@glxtr@s@longnewglossaryentry}[3]{%
641   \glsdofnoexists{#1}%
642   {%
643     \bgroup
644     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
645     \long\def\@newglossaryentryprehook{%
646       \long\def\@glo@desc{#3}%
647       \@org@newglossaryentryprehook
648     }%
649     \renewcommand*\@gls@assign@desc}[1]{%
650       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
651       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
652     }
653     \gls@defglossaryentry{#1}{#2}%
654   \egroup
655 }%
656 }
```

ewglossaryentry Unstarred version.

```
657 \newcommand{\@glxtr@longnewglossaryentry}[3]{%
658   \glsdofnoexists{#1}%
659   {%
660     \bgroup
661     \let\@org@newglossaryentryprehook\@newglossaryentryprehook
662     \long\def\@newglossaryentryprehook{%
663       \long\def\@glo@desc{#3\glxtrpostlongdescription}%
664       \@org@newglossaryentryprehook
665     }%
666     \renewcommand*\@gls@assign@desc}[1]{%
667       \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
668       \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
669     }
670     \gls@defglossaryentry{#1}{#2}%
671   \egroup
672 }%
673 }
```

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

```
674 \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.

```
675 \renewcommand{\newignoredglossary}{%
676   \@ifstar\glxtr@s@newignoredglossary\glxtr@org@newignoredglossary
677 }
```


ignoredglossary The original definition is patched to check for existence.

```
678 \newcommand*{\glxtr@org@newignoredglossary}[1]{%
679   \ifcsdef{glolist@#1}
680   {%
681     \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
682   }%
683   {%
684     \ifdefempty\@ignored@glossaries
685     {%
686       \edef\@ignored@glossaries{#1}%
687     }%
688     {%
689       \eappto\@ignored@glossaries{, #1}%
690     }%
691     \csgdef{glolist@#1}{,}%
692     \ifcsundef{gls@#1@entryfmt}%
693     {%
694       \defglsentryfmt[#1]{\glsentryfmt}%
695     }%
696     {}%
697     \ifdefempty\@gls@nohyperlist
698     {%
699       \renewcommand*{\@gls@nohyperlist}{#1}%
700     }%
701     {%
702       \eappto\@gls@nohyperlist{, #1}%
703     }%
704   }%
705 }
```

ignoredglossary Starred form.

```
706 \newcommand*{\glxtr@s@newignoredglossary}[1]{%
707   \ifcsdef{glolist@#1}
708   {%
709     \glxtrundefaction{Glossary type ‘#1’ already exists}{}%
710   }%
711   {%
712     \ifdefempty\@ignored@glossaries
713     {%
714       \edef\@ignored@glossaries{#1}%
715     }%
716     {%
717       \eappto\@ignored@glossaries{, #1}%
718     }%
719     \csgdef{glolist@#1}{,}%
720     \ifcsundef{gls@#1@entryfmt}%
721     {%
722       \defglsentryfmt[#1]{\glsentryfmt}%
723     }%
724   }
```

```

724     {}%
725   }%
726 }

```

`\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.

```

727 \glsifusetranslator
728 {%
729   \renewcommand*{\glsettoctitle}[1]{%
730     \ifcsdef{gls@tr@set@#1@toctitle}%
731       {%
732         \csuse{gls@tr@set@#1@toctitle}%
733       }%
734     {%
735       \ifcsdef{@glotype@#1@title}%
736         {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
737         {\def\glossarytoctitle{\glossarytitle}}%
738       }%
739     }%
740 }
741 {
742   \renewcommand*{\glsettoctitle}[1]{%
743     \ifcsdef{@glotype@#1@title}%
744       {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
745       {\def\glossarytoctitle{\glossarytitle}}%
746   }
747 }

```

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

```

748 \newcommand{\provideignoredglossary}{%
749   \@ifstar{glsxtr@s@provideignoredglossary}{glsxtr@provideignoredglossary}
750 }

```

`ignoredglossary` Unstarred version.

```

751 \newcommand*{\glsxtr@provideignoredglossary}[1]{%
752   \ifcsdef{glolist@#1}
753     {}%
754     {%
755       \ifdefempty\@ignored@glossaries
756         {%
757           \edef\@ignored@glossaries{#1}%
758         }%
759         {%
760           \eappto\@ignored@glossaries{,#1}%
761         }%
762       \csgdef{glolist@#1}{,}%
763       \ifcsundef{gls@#1@entryfmt}%
764         {%
765           \defglsentryfmt[#1]{\glsentryfmt}%

```

```

766 }%
767 {}%
768 \ifdefempty\@gls@nohyperlist
769 {%
770   \renewcommand*\@gls@nohyperlist{#1}%
771 }%
772 {%
773   \eappto\@gls@nohyperlist{, #1}%
774 }%
775 }%
776 }

```

`ignoredglossary` Starred form.

```

777 \newcommand*\@glsxtr@s@provideignoredglossary}[1]{%
778   \ifcsdef{glolist@#1}
779   {}%
780   {%
781     \ifdefempty\@ignored@glossaries
782     {%
783       \edef\@ignored@glossaries{#1}%
784     }%
785     {%
786       \eappto\@ignored@glossaries{, #1}%
787     }%
788     \csgdef{glolist@#1}{,}%
789     \ifcsundef{gls@#1@entryfmt}%
790     {%
791       \defglsentryfmt[#1]{\glsentryfmt}%
792     }%
793     {}%
794   }%
795 }

```

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

```

796 \newcommand*\@glsxtrcopytoglossary}[2]{%
797   \glsdoifexists{#1}%
798   {%
799     \ifcsdef{glolist@#2}
800     {%
801       \cseappto{glolist@#2}{#1,}%
802     }%
803     {%
804       \glsxtrundefaction{Glossary type ‘#2’ doesn’t exist}{}%
805     }%
806   }%
807 }

```

1.3.1 Existence Checks

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

```
808 \renewcommand{\glsdoifexists}[2]{%
809   \ifglstryexists{#1}{#2}%
810   {%
      Define \glslabel in case it's needed after this command (for example in the post-link hook).

811   \edef\glslabel{\glsdetoklabel{#1}}%
812   \glstrundefaction{Glossary entry '\glslabel'
813     has not been defined}{You need to define a glossary entry before
814     you can reference it.}%
815   }%
816 }
```

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

```
817 \renewcommand{\glsdoifnoexists}[2]{%
818   \ifglstryexists{#1}{%
819     \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
820       has already been defined}{}}{#2}%
821 }
```

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

```
822 \ifdef\glsdoifexistsordo
823 {%
824   \renewcommand{\glsdoifexistsordo}[3]{%
825     \ifglstryexists{#1}{#2}%
826     {%
827       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
828         has not been defined}{You need to define a glossary entry
829         before you can use it.}%
830       #3%
831     }%
832   }%
833 }
834 {%
835   \glstr@warnonexistsordo\glsdoifexistsordo
836   \newcommand{\glsdoifexistsordo}[3]{%
837     \ifglstryexists{#1}{#2}%
838     {%
839       \glstrundefaction{Glossary entry '\glsdetoklabel{#1}'
840         has not been defined}{You need to define a glossary entry
841         before you can use it.}%
842       #3%
843     }%
844   }%
845 }
```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```
846 \ifdef\doifglossarynoexistsordo
847 {%
848   \renewcommand{\doifglossarynoexistsordo}[3]{%
849     \ifglossaryexists{#1}%
850     {%
851       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
852       #3%
853     }%
854     {#2}%
855   }%
856 }
857 {%
858   \glstr@warnonexistsordo\doifglossarynoexistsordo
859   \newcommand{\doifglossarynoexistsordo}[3]{%
860     \ifglossaryexists{#1}%
861     {%
862       \glstrundefaction{Glossary type ‘#1’ already exists}{}%
863       #3%
864     }%
865     {#2}%
866   }%
867 }
868
```

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

```
869 \appto\@newglossaryentryposthook{%
870   \ifdefvoid\@glo@see
871   {\csxdef{glo@\@glo@label @see}{}}%
872   {%
873     \csxdef{glo@\@glo@label @see}{\@glo@see}%
874     \@glstr@autoindexcrossrefs
875   }%
876 }
877 \appto\@gls@keymap{,{see}{see}}
```

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

```
878 \newcommand*{\glstrusesee}[1]{%
879   \glsdoifexists{#1}%
880   {%
881     \letcs{\@glo@see}{glo\@glsdetoklabel{#1}@see}%
882     \ifdefempty\@glo@see
883     {}%
884     {%
885       \expandafter\glstr@usesee\@glo@see\@end@glstr@usesee
886     }%
887   }%
888 }
```

```

\glxtr@usesee
889 \newcommand*{\glxtr@usesee}[1][\seename]{%
890   \@glxtr@usesee[#1]%
891 }

\@glxtr@usesee
892 \def\@glxtr@usesee[#1]#2\@end@glxtr@usesee{%
893   \glxtruseseeformat{#1}{#2}%
894 }

xtruseseeformat  The format used by \glxtrusesee. The first argument is the tag (such as \seename). The
                  second argument is the comma-separated list of cross-referenced labels.
895 \newcommand*{\glxtruseseeformat}[2]{%
896   \glssseeformat{#1}{#2}{}%
897 }

                  Add all unused cross-references at the end of the document.
898 \AtEndDocument{\if@glxtrindexcrossrefs\glxtraddallcrossrefs\fi}

addallcrossrefs  Iterate through all used entries and if they have a cross-reference, make sure the cross-
                  reference has been added.
899 \newcommand*{\glxtraddallcrossrefs}{%
900   \forallglossaries{\@glo@type}%
901   {%
902     \forlentries[\@glo@type]{\@glo@label}%
903     {%
904       \ifglused{\@glo@label}{\@glxtr@addunusedxrefs{\@glo@label}}{}}%
905     }%
906   }%
907 }

@addunusedxrefs  If the given entry has a see field add all unused cross-references.
908 \newcommand*{\@glxtr@addunusedxrefs}[1]{%
909   \letcs{\@glo@see}{glo@glstoklabel{#1}@see}%
910   \ifdefvoid\@glo@see
911   {%
912   {%
913     \expandafter\glxtr@addunused\@glo@see\@end@glxtr@addunused
914   }%
915 }

lsxtr@addunused  Adds all the entries if they haven't been used.
916 \newcommand*{\glxtr@addunused}[1][{}]{%
917   \@glxtr@addunused
918 }

lsxtr@addunused  Adds all the entries if they haven't been used.
919 \def\@glxtr@addunused#1\@end@glxtr@addunused{%

```

```

920 \@for\@glxstr@label:=#1\do
921 {%
922   \ifglxsused{\@glxstr@label}{}%
923   {%
924     \glxsadd[format=glxstrunusedformat]{\@glxstr@label}%
925     \glxsunset{\@glxstr@label}%
926     \@glxstr@addunusedxrefs{\@glxstr@label}%
927   }%
928 }%
929 }

```

glxstrunusedformat

```

930 \newcommand*{\glxstrunusedformat}[1]{\unskip}

```

1.3.2 Document Definitions

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

```

931 \let\glxstr@orgmakenoidxglossaries\makenoidxglossaries
932 \renewcommand{\makenoidxglossaries}{%
933   \glxstr@orgmakenoidxglossaries
934   \if@glxstrdocdefrestricted

```

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

```

935   \renewcommand*{\@gls@reference}[3]{%
936     \ifcsundef{@glsref@##1}{\csgdef{@glsref@##1}{}}{}%
937     \ifinlistcs{##2}{@glsref@##1}%
938     {}%
939     {\listcsgadd{@glsref@##1}{##2}}%
940     \ifcsundef{glo@glsdetoklabel{##2}@loclist}%
941     {\csgdef{glo@glsdetoklabel{##2}@loclist}{}}{}%
942     {}%
943     \listcsgadd{glo@glsdetoklabel{##2}@loclist}{##3}%
944   }%
945   \else

```

Disable document definitions.

```

946   \@glxstrdocdeffalse
947   \fi
948   \disable@keys{glossaries-extra}{docdef}%
949 }

```

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

```

950 \renewcommand*{\gls@defdocnewglossaryentry}{%
951   \ifcase\@glxstr@docdefval
952   docdef=false:
953   \renewcommand*{\newglossaryentry}[2]{%
954     \PackageError{glossaries-extra}{Glossary entries must

```

```

954     be \MessageBreak defined in the preamble with \MessageBreak
955     package option 'docdef=false'\MessageBreak(consider using
956     'docdef=restricted')){Move your glossary definitions to
957     the preamble. You can also put them in a \MessageBreak separate file
958     and load them with \string\loadglsentries.}%
959 }%

```

```

960 \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.

```

961 \let\gls@checkseeallowed\relax
962 \let\newglossaryentry\new@glossaryentry
963 \or

```

Restricted mode just needs to allow the see value.

```

964 \let\gls@checkseeallowed\relax
965 \fi
966 }%

```

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

```

967 \newcommand*{\GlsXtrEnableOnTheFly}{%
968 \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
969 }

```

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.)

```

970 \newcommand*{\@sGlsXtrEnableOnTheFly}{%
971 \renewcommand*{\glsdetoklabel}[1]{%
972 \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
973 {%
974 \expandafter\detokenize\expandafter{##1}%
975 }%
976 {\detokenize{##1}}%
977 }%
978 \@GlsXtrEnableOnTheFly
979 }
980 \def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
981 \expandafter\if\glsbackslash#1%
982 #3%
983 \else
984 #4%
985 \fi
986 }

```


sxtrstarflywarn

```
987 \newcommand*{\glxsxtrstarflywarn}{%
988   \GlossariesExtraWarning{Experimental starred version of
989   \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
990   read the warnings in the glossaries-extra user manual)}}%
991 }
```

rEnableOnTheFly

```
992 \newcommand*{\@GlsXtrEnableOnTheFly}{%
```

Don't redefine `\glstdetoklabel` 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`

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

`\glsxtr`

```
994   \newcommand*{\glsxtr}[1] [] {%
995     \def\glsxtr@keylist{##1}%
996     \@glsxtr
997   }
```

`\@glsxtr`

```
998   \newcommand*{\@glsxtr}[2] [] {%
999     \ifglstryexists{##2}%
1000     {%
1001       \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1002     }%
1003     {%
1004       \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1005       description={\nopostdesc},##1}%
1006     }%
1007     \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
1008   }
```

`\Glsxtr`

```
1009   \newcommand*{\Glsxtr}[1] [] {%
1010     \def\glsxtr@keylist{##1}%
1011     \@Glsxtr
1012   }
```

`\@Glsxtr`

```
1013   \newcommand*{\@Glsxtr}[2] [] {%
1014     \ifglstryexists{##2}%
1015     {%
1016       \ifblank{##1}{-}{\GlsXtrWarning{##1}{##2}}%
1017     }%
```

```

1018  {%
1019    \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1020      description={\nopostdesc},##1}%
1021  }%
1022  \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
1023  }

```

\glsxtrpl

```

1024  \newcommand*{\glsxtrpl}[1][]{%
1025    \def\glsxtr@keylist{##1}%
1026    \@glsxtrpl
1027  }

```

\@glsxtrpl

```

1028  \newcommand*{\@glsxtrpl}[2][]{%
1029    \ifglsentryexists{##2}%
1030    {%
1031      \ifblank{##1}{ }\{\GlsXtrWarning{##1}{##2}}%
1032    }%
1033    {%
1034      \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1035        description={\nopostdesc},##1}%
1036    }%
1037    \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
1038  }

```

\Glsxtrpl

```

1039  \newcommand*{\Glsxtrpl}[1][]{%
1040    \def\glsxtr@keylist{##1}%
1041    \@Glsxtrpl
1042  }

```

\@Glsxtrpl

```

1043  \newcommand*{\@Glsxtrpl}[2][]{%
1044    \ifglsentryexists{##2}
1045    {%
1046      \ifblank{##1}{ }\{\GlsXtrWarning{##1}{##2}}%
1047    }%
1048    {%
1049      \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
1050        description={\nopostdesc},##1}%
1051    }%
1052    \expandafter\Glspl\expandafter[\glsxtr@keylist]{##2}%
1053  }

```

\GlsXtrWarning

```

1054  \newcommand*{\GlsXtrWarning}[2]{%
1055    \def\@glsxtr@optlist{##1}%
1056    \@onelevel@sanitize\@glsxtr@optlist

```

```

1057 \GlossariesExtraWarning{The options ‘\@glsxtr@optlist’ have
1058 been ignored for entry ‘##2’ as it has already been defined}%
1059 }

```

Disable commands after the glossary:

```

1060 \renewcommand\@printglossary[2]{%
1061 \def\@glsxtr@printglossopts{##1}%
1062 \@glsxtr@orgprintglossary{##1}{##2}%
1063 \def\@glsxtr{\@glsxtr@disabledflycommand\glsxtr}%
1064 \def\@glsxtrpl{\@glsxtr@disabledflycommand\glsxtrpl}%
1065 \def\@Glsxtr{\@glsxtr@disabledflycommand\Glsxtr}%
1066 \def\@Glsxtrpl{\@glsxtr@disabledflycommand\Glsxtrpl}%
1067 }

```

abledflycommand

```

1068 \newcommand*{\@glsxtr@disabledflycommand}[1]{%
1069 \PackageError{glossaries-extra}%
1070 {\string##1\space can’t be used after any of the \MessageBreak
1071 glossaries have been displayed}%
1072 {The on-the-fly commands enabled by
1073 \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
1074 before the glossaries. If you want to use any entries \MessageBreak
1075 after any of the glossaries, you must use the standard \MessageBreak
1076 method of first defining the entry and then using the \MessageBreak
1077 entry with commands like \string\gls}%
1078 \@glsxtr@disabledflycommand
1079 }%
1080 \newcommand*{\@glsxtr@disabledflycommand}[2][{}]{##2}

```

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

```

1081 \let\GlsXtrEnableOnTheFly\relax
1082 }
1083 \@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.

```

1084 \newcommand*{\@glsxtr@current@style}{\@glossary@default@style}

```

Modify \setglossarystyle to set the above.

etglossarystyle

```

1085 \renewcommand*{\setglossarystyle}[1]{%
1086 \ifcsundef{@glsstyle@#1}%
1087 {%
1088 \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%

```

```

1089 }%
1090 {%
1091   \csname @glsstyle@#1\endcsname

```

Only set the current style if it exists.

```

1092   \protected@edef\@glstr@current@style{#1}%
1093 }%
1094 \ifx\@glossary@default@style\relax
1095   \protected@edef\@glossary@default@style{#1}%
1096 \fi
1097 }

```

In case we have an old version of glossaries:

```

1098 \ifdef\@glossary@default@style
1099 {}
1100 {%
1101   \let\@glossary@default@style\relax
1102 }

```

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

```

1103 \ifdef\glslistdottedwidth
1104 {%
1105   \ifdim\glslistdottedwidth=.5\hsize
1106     \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
1107     \AtBeginDocument{%
1108       \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
1109         \setlength{\glslistdottedwidth}{.5\columnwidth}%
1110       \fi
1111     }%
1112 \fi
1113 }
1114 {}%

```

Similarly for `\glsdescwidth`:

`\glsdescwidth`

```

1115 \ifdef\glsdescwidth
1116 {%
1117   \ifdim\glsdescwidth=.6\hsize
1118     \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
1119     \AtBeginDocument{%
1120       \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
1121         \setlength{\glsdescwidth}{.6\columnwidth}%
1122       \fi
1123     }%
1124 \fi
1125 }
1126 {}%

```

and for `\glspagelistwidth`:

`lspagelistwidth`

```
1127 \ifdef\glspagelistwidth
1128 {%
1129   \ifdim\glspagelistwidth=.1\hsize
1130     \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
1131     \AtBeginDocument{%
1132       \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
1133         \setlength{\glspagelistwidth}{.1\columnwidth}%
1134       \fi
1135     }%
1136   \fi
1137 }
1138 {}%
```

`aryentrynumbers` Has the nonumberlist option been used?

```
1139 \def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
1140 \ifx\org@glossaryentrynumbers\glossaryentrynumbers
1141   \glsnonumberlistfalse
1142   \renewcommand*{\glossaryentrynumbers}[1]{%
1143     \ifglentryexists{\glscurrententrylabel}%
1144     {%
1145       \@glsxtrpreloctag
1146       \GlsXtrFormatLocationList{#1}%
1147       \@glsxtrpostloctag
1148       \gls@save@numberlist{#1}%
1149     }{}%
1150   }%
1151 \else
1152   \glsnonumberlisttrue
1153   \renewcommand*{\glossaryentrynumbers}[1]{%
1154     \ifglentryexists{\glscurrententrylabel}%
1155     {%
1156       \gls@save@numberlist{#1}%
1157     }{}%
1158   }%
1159 \fi
```

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

```
1160 \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

```
1161 \newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
1162   \let\@glxstrpreloctag\@glxstrpreloctag
1163   \let\@glxstrpostloctag\@glxstrpostloctag
1164   \renewcommand*{\@glxstr@pagetag}{#1}%
1165   \renewcommand*{\@glxstr@pagetag}{#2}%
1166   \renewcommand*{\@glxstr@savepreloctag}[2]{%
1167     \csgdef{\@glxstr@preloctag##1}{##2}%
1168   }%
1169   \renewcommand*{\@glxstr@doloctag}{%
1170     \ifcsundef{\@glxstr@preloctag\@glscurrententrylabel}%
1171     {%
1172       \GlossariesWarning{Missing pre-location tag for ‘\@glscurrententrylabel’.
1173       Rerun required}%
1174     }%
1175     {%
1176       \csuse{\@glxstr@preloctag\@glscurrententrylabel}%
1177     }%
1178   }%
1179 }
1180 \@onlypreamble\GlsXtrEnablePreLocationTag
```

glxstrpreloctag

```
1181 \newcommand*{\@glxstrpreloctag}{%
1182   \let\@glxstr@org@delimN\delimN
1183   \let\@glxstr@org@delimR\delimR
1184   \let\@glxstr@org@glsgignore\glsgignore
1185   \gdef is required as the delimiters may occur inside a scope.
1186   \gdef\@glxstr@thisloctag{\@glxstr@pagetag}%
1187   \renewcommand*{\delimN}{%
1188     \gdef\@glxstr@thisloctag{\@glxstr@pagetag}%
1189     \@glxstr@org@delimN}%
1190   \renewcommand*{\delimR}{%
1191     \gdef\@glxstr@thisloctag{\@glxstr@pagetag}%
1192     \@glxstr@org@delimR}%
1193   \renewcommand*{\glsgignore}[1]{%
1194     \gdef\@glxstr@thisloctag{\relax}%
1195     \@glxstr@org@glsgignore{##1}}%
1196   \@glxstr@doloctag
1197 }
```

glxstrpreloctag

```
1197 \newcommand*{\@glxstrpreloctag}{}
```

@glxstr@pagetag

```
1198 \newcommand*{\@glxstr@pagetag}{}
```

glxstr@pagetag

```
1199 \newcommand*{\@glxstr@pagetag}{}
```

lsxtrpostloctag

```
1200 \newcommand*{\@@glxtrpostloctag}{%
1201   \let\delimN\@glxtr@org@delimN
1202   \let\delimR\@glxtr@org@delimR
1203   \let\glignore\@glxtr@org@glignore
1204   \protected@write\@auxout{}{%
1205     {\string\@glxtr@savepreloctag{\glscurrententrylabel}{\@glxtr@thisloctag}}%
1206 }
```

lsxtrpostloctag

```
1207 \newcommand*{\@glxtrpostloctag}{}
```

lsxtr@preloctag

```
1208 \newcommand*{\@glxtr@savepreloctag}[2]{%
1209 \protected@write\@auxout{}{%
1210   \string\providecommand\string\@glxtr@savepreloctag[2]{}}
```

glxtr@doloctag

```
1211 \newcommand*{\@glxtr@doloctag}{}
```

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

```
1212 \renewcommand*{\KV@printgloss@nonumberlist}[1]{%
1213   \XKV@plfalse
1214   \XKV@sttrue
1215   \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
1216   {%
1217     \csname glsnonumberlist\XKV@resa\endcsname
1218     \ifglsnonumberlist
1219       \def\glossaryentrynumbers##1{\gl@save@numberlist{##1}}%
1220     \else
1221       \def\glossaryentrynumbers##1{%
1222         \@glxtrpreloctag
1223         \GlsXtrFormatLocationList{##1}%
1224         \@glxtrpostloctag
1225         \gl@save@numberlist{##1}}%
1226     \fi
1227   }%
1228 }
```

1.3.4 Entry Formatting, Hyperlinks and Indexing

`\glentryfmt` 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 `\glentryfmt` will need redefining as appropriate (or use `\defglentryfmt`). 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.

```

1229 \renewcommand*{\glsentryfmt}{%
1230   \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}{}}{%
1231     \glsifregular{\glslabel}%
1232     {\glsxtrregularfont{\glsentryfmt}}%
1233     {%
1234       \ifglshasshort{\glslabel}%
1235       {\glsxtrgenabbrvfmt}%
1236       {\glsxtrregularfont{\glsentryfmt}}%
1237     }%
1238 }

```

sxtrregularfont Font used for regular entries.

```

1239 \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.

```

1240 \renewcommand{\@gls@field@link}[4][]{%

```

If the `record` option has been used, the information needs to be written to the aux file regardless of whether the entry exists.

```

1241   \@glsxtr@record{#2}{#3}{\glslink}%
1242   \glsdoifexists{#3}%
1243   {%

```

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).

```

1244   \let\glsxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
1245   \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1246   \def\glscustomtext{#4}%
1247   \@glsxtr@field@linkdefs
1248   #1%
1249   \@gls@link[#2]{#3}{#4}%
1250   \let\ifKV@glslink@hyper\glsxtrorg@ifKV@glslink@hyper
1251   }%
1252   \glspostlinkhook
1253 }

```

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.

```

1254 \let\@glsxtr@org@gls@\@gls@
1255 \def\@gls@#1#2{%

```



```

1256 \@glstr@record{#1}{#2}{glslink}%
1257 \@glstr@org@glstr@{#1}{#2}%
1258 }%

```

`\@glsp1@` Save the original definition and redefine.

```

1259 \let\@glstr@org@glsp1@\@glsp1@
1260 \def\@glsp1@#1#2{%
1261   \@glstr@record{#1}{#2}{glslink}%
1262   \@glstr@org@glsp1@{#1}{#2}%
1263 }%

```

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

```

1264 \let\@glstr@org@Gls@\@Gls@
1265 \def\@Gls@#1#2{%
1266   \@glstr@record{#1}{#2}{glslink}%
1267   \@glstr@org@Gls@{#1}{#2}%
1268 }%

```

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

```

1269 \let\@glstr@org@Glspl@\@Glspl@
1270 \def\@Glspl@#1#2{%
1271   \@glstr@record{#1}{#2}{glslink}%
1272   \@glstr@org@Glspl@{#1}{#2}%
1273 }%

```

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

```

1274 \let\@glstr@org@GLS@\@GLS@
1275 \def\@GLS@#1#2{%
1276   \@glstr@record{#1}{#2}{glslink}%
1277   \@glstr@org@GLS@{#1}{#2}%
1278 }%

```

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

```

1279 \let\@glstr@org@GLSpl@\@GLSpl@
1280 \def\@GLSpl@#1#2{%
1281   \@glstr@record{#1}{#2}{glslink}%
1282   \@glstr@org@GLSpl@{#1}{#2}%
1283 }%

```

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

```

1284 \renewcommand*{\@glsdisp}[3][{}]{%
1285   \@glstr@record{#1}{#2}{glslink}%
1286   \glsdoifexists{#2}{%
1287     \let\do@glstr@link@checkfirsthyper\@glstr@link@checkfirsthyper
1288     \let\glstr@ifplural\@secondoftwo
1289     \let\glstr@scase\@firstofthree
1290     \def\glstr@customtext{#3}%

```

```

1291 \def\glsinsert{}%
1292 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
1293 \@gls@link[#1]{#2}{\@glo@text}%
1294 \ifKV@glslink@local
1295 \glslocalunset{#2}%
1296 \else
1297 \glsunset{#2}%
1298 \fi
1299 }%
1300 \glspostlinkhook
1301 }

```

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

```

1302 \renewcommand*{\@gls@@link}[3][{}]{%
1303 \@glxtr@record{#1}{#2}{glslink}%
1304 \glsdoifexistsordo{#2}%
1305 {%
1306 \let\do@gls@link@checkfirsthyper\relax
1307 \@gls@link[#1]{#2}{#3}%
1308 }%
1309 {%
1310 \glstextformat{#3}%
1311 }%
1312 \glspostlinkhook
1313 }

```

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

```

1314 \newcommand*{\glxtrinitwrgloss}{%
1315 \glsifattribute{\glslabel}{wrgloss}{after}%
1316 {%
1317 \glxtrinitwrglossbeforefalse
1318 }%
1319 {%
1320 \glxtrinitwrglossbeforetrue
1321 }%
1322 }

```

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

```

1323 \newif\ifglxtrinitwrglossbefore
1324 \glxtrinitwrglossbeforetrue

```

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

```

1325 \define@choicekey{glslink}{wrgloss}[\val\nr]{before,after}%
1326 {%
1327 \ifcase\nr\relax
1328 \glxtrinitwrglossbeforetrue
1329 \or
1330 \glxtrinitwrglossbeforefalse

```

```

1331 \fi
1332 }

```

`\@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`.

```

1333 \def\@gls@link[#1]#2#3{%
1334   \leavevmode
1335   \edef\glslabel{\glsdetoklabel{#2}}%
1336   \def\@gls@link@opts{#1}%
1337   \let\@gls@link@label\glslabel
1338   \def\@glsnumberformat{glsnumberformat}%
1339   \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
1340   \edef\gls@type{\csname glo@\glslabel @type\endcsname}%
1341   \let\org@ifKV@glslink@hyper@ifKV@glslink@hyper

```

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

```

1342 \glsxtrinitwrgloss

```

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

```

1343 \@gls@setdefault@glslink@opts
1344 \do@glsdisablehyperinlist
1345 \do@gls@link@checkfirsthyper
1346 \setkeys{glslink}{#1}%
1347 \glslinkpostsetkeys
1348 \@gls@saveentrycounter
1349 \@gls@setsort{\glslabel}%

```

Do write if it should occur before the link text:

```

1350 \ifglsxtrinitwrglossbefore
1351   \do@wrglossary{#2}%
1352 \fi

```

Do the link text:

```

1353 \ifKV@glslink@hyper
1354   \@glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
1355 \else
1356   \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
1357 \fi

```

Do write if it should occur after the link text:

```

1358 \ifglsxtrinitwrglossbefore
1359 \else
1360   \do@wrglossary{#2}%
1361 \fi

```

As the original definition:

```

1362 \let@ifKV@glslink@hyper\org@ifKV@glslink@hyper
1363 }

```

```

1364 \define@key{glossadd}{thevalue}{\def\@glsxtr@thevalue{#1}}

```

```
1365 \define@key{glossadd}{theHvalue}{\def\@glxtr@theHvalue{#1}}
```

\glsadd Redefine to include \@glxtr@record

```
1366 \renewrobustcmd*{\glsadd}[2][]{%
1367   \@gls@adjustmode
1368   \@glxtr@record{#1}{#2}{glossadd}%
1369   \glsdoifexists{#2}%
1370   {%
1371     \def\@glsnumberformat{glsnumberformat}%
1372     \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
1373     \def\@glxtr@thevalue{%
1374       \def\@glxtr@theHvalue{\@glxtr@thevalue}%
1375       \setkeys{glossadd}{#1}%
1376       \ifdefempty{\@glxtr@thevalue}%
1377       {%
1378         \@gls@saveentrycounter
1379       }%
1380       {%
1381         \let\theglentrycounter\@glxtr@thevalue
1382         \def\theHglentrycounter{\@glxtr@theHvalue}%
1383       }%
1384       \@do@wrglossary{#2}%
1385     }%
1386 }
```

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

```
1387 \newcommand*{\@glxtr@field@linkdefs}{%
1388   \let\glxtrifwasfirstuse\@secondoftwo
1389   \let\glsifplural\@secondoftwo
1390   \let\glscapscase\@firstofthree
1391   \let\glsinsert\@empty
1392 }
```

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

assignfieldfont

```
1393 \newcommand*{\glxtrassignfieldfont}[1]{%
1394   \ifglentryexists{#1}%
1395   {%
1396     \ifglshasshort{#1}%
1397     {%
1398       \glsetabbrvfmt{\glscategory{#1}}%
1399       \glsifregular{#1}%
1400       {\let\@gls@field@font\glxtrregularfont}%
1401       {\let\@gls@field@font\@firstofone}%
1402     }%
1403     {%
1404       \glsifnotregular{#1}%
1405     }
```

```

1405     {\let\@gls@field@font\@firstofone}%
1406     {\let\@gls@field@font\glsxtrregularfont}%
1407   }%
1408 }%
1409 {%
1410   \let\@gls@field@font\@gobble
1411 }%
1412 }

```

`\@gls@text@` The abbreviation format may also need setting.

```

1413 \def\@gls@text@#1#2[#3]{%
1414   \glsxtrassignfieldfont{#2}%
1415   \@gls@field@link{#1}{#2}{\@gls@field@font{\gls@text@{#2}#3}}%
1416 }

```

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

```

1417 \def\@GLStext@#1#2[#3]{%
1418   \glsxtrassignfieldfont{#2}%
1419   \@gls@field@link[\let\gls@capscase\@thirdofthree]{#1}{#2}%
1420   {\@gls@field@font{\GLS@text@{#2}\mfirstucMakeUppercase{#3}}}%
1421 }

```

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

```

1422 \def\@Gls@text@#1#2[#3]{%
1423   \glsxtrassignfieldfont{#2}%
1424   \@gls@field@link[\let\gls@capscase\@secondofthree]{#1}{#2}%
1425   {\@gls@field@font{\Gls@text@{#2}#3}}%
1426 }

```

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.

`\@gls@checknohyperfirst`

```

1427 \newcommand*\@gls@checknohyperfirst[1]{%
1428   \gls@ifattribute{#1}{nohyperfirst}{true}{\KV@gls@link@hyperfalse}{}}%
1429 }

```

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

```

1430 \def\@glsfirst@#1#2[#3]{%
1431   \glsxtrassignfieldfont{#2}%
1432   \@gls@field@link
1433   [\let\glsxtrifwasfirstuse\@firstoftwo
1434   \glsxtrchecknohyperfirst{#2}%
1435   ]{#1}{#2}%
1436   {\@gls@field@font{\gls@first@{#2}#3}}%
1437 }

```

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

```
1438 \def\@Glsfirst@#1#2[#3]{%
1439   \glstrassignfieldfont{#2}%
      Ensure that \Glsfirst honours the nohyperfirst attribute.
1440   \@gls@field@link
1441   [\let\glstrifwasfirstuse\@firstoftwo
1442     \let\glscapscase\@secondofthree
1443     \glstrchecknohyperfirst{#2}%
1444   ]%
1445   {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
1446 }
```

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

```
1447 \def\@GLSfirst@#1#2[#3]{%
1448   \glstrassignfieldfont{#2}%
      Ensure that \GLSfirst honours the nohyperfirst attribute.
1449   \@gls@field@link
1450   [\let\glstrifwasfirstuse\@firstoftwo
1451     \let\glscapscase\@thirdofthree
1452     \glstrchecknohyperfirst{#2}%
1453   ]%
1454   {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
1455 }
```

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

```
1456 \def\@glsplural@#1#2[#3]{%
1457   \glstrassignfieldfont{#2}%
1458   \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
1459   {\@gls@field@font{\glsaccessplural{#2}#3}}%
1460 }
```

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

```
1461 \def\@Glsplural@#1#2[#3]{%
1462   \glstrassignfieldfont{#2}%
1463   \@gls@field@link
1464   [\let\glsifplural\@firstoftwo
1465     \let\glscapscase\@secondofthree
1466   ]%
1467   {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
1468 }
```

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

```
1469 \def\@GLSplural@#1#2[#3]{%
1470   \glstrassignfieldfont{#2}%
1471   \@gls@field@link
1472   [\let\glsifplural\@firstoftwo
1473     \let\glscapscase\@thirdofthree
```

```

1474 ]%
1475 {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
1476 }

```

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

```

1477 \def\@glsfirstplural@#1#2[#3]{%
1478   \glstrassignfieldfont{#2}%
    Ensure that \glsfirstplural honours the nohyperfirst attribute.
1479   \@gls@field@link
1480   [\let\glstrifwasfirstuse\@firstoftwo
1481     \let\glsifplural\@firstoftwo
1482     \glstrchecknohyperfirst{#2}%
1483   ]%
1484   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
1485 }

```

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

```

1486 \def\@Glsfirstplural@#1#2[#3]{%
1487   \glstrassignfieldfont{#2}%
    Ensure that \glsfirstplural honours the nohyperfirst attribute.
1488   \@gls@field@link
1489   [\let\glstrifwasfirstuse\@firstoftwo
1490     \let\glsifplural\@firstoftwo
1491     \let\glscapscase\@secondofthree
1492     \glstrchecknohyperfirst{#2}%
1493   ]%
1494   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
1495 }

```

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

```

1496 \def\@GLSfirstplural@#1#2[#3]{%
1497   \glstrassignfieldfont{#2}%
    Ensure that \glsfirstplural honours the nohyperfirst attribute.
1498   \@gls@field@link
1499   [\let\glstrifwasfirstuse\@firstoftwo
1500     \let\glsifplural\@firstoftwo
1501     \let\glscapscase\@thirdofthree
1502     \glstrchecknohyperfirst{#2}%
1503   ]%
1504   {#1}{#2}%
1505   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
1506 }

```

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

```

1507 \def\@glsname@#1#2[#3]{%
1508   \glstrassignfieldfont{#2}%
1509   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}#3}}%
1510 }

```

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

```
1511 \def\@Glsname@#1#2[#3]{%
1512   \glstrassignfieldfont{#2}%
1513   \@gls@field@link
1514   [\let\glscapscase\@secondoftwo]{#1}{#2}%
1515   {\@gls@field@font{\Glsaccessname{#2}#3}}%
1516 }
```

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

```
1517 \def\@GLSname@#1#2[#3]{%
1518   \glstrassignfieldfont{#2}%
1519   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1520   {#1}{#2}%
1521   {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
1522 }
```

\@glsdesc@

```
1523 \def\@glsdesc@#1#2[#3]{%
1524   \glstrassignfieldfont{#2}%
1525   \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
1526 }
```

\@Glsdesc@ First letter uppercase version.

```
1527 \def\@Glsdesc@#1#2[#3]{%
1528   \glstrassignfieldfont{#2}%
1529   \@gls@field@link
1530   [\let\glscapscase\@secondoftwo]{#1}{#2}%
1531   {\@gls@field@font{\Glsaccessdesc{#2}#3}}%
1532 }
```

\@GLSdesc@ All uppercase version.

```
1533 \def\@GLSdesc@#1#2[#3]{%
1534   \glstrassignfieldfont{#2}%
1535   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1536   {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
1537 }
```

@glsdescplural@ No case-changing version.

```
1538 \def\@glsdescplural@#1#2[#3]{%
1539   \glstrassignfieldfont{#2}%
1540   \@gls@field@link
1541   [\let\glscapscase\@secondoftwo
1542    \let\glsifplural\@firstoftwo
1543   ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
1544 }
```

@Glsdescplural@ First letter uppercase version.

```
1545 \def\@Glsdescplural@#1#2[#3]{%
```



```

1546 \glstrassignfieldfont{#2}%
1547 \@gls@field@link
1548 [\let\glscapscase\@secondoftwo
1549 \let\glsifplural\@firstoftwo
1550 ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
1551 }

```

@GLSdescplural@ All uppercase version.

```

1552 \def\@GLSdesc@#1#2[#3]{%
1553 \glstrassignfieldfont{#2}%
1554 \@gls@field@link
1555 [\let\glscapscase\@thirdoftwo
1556 \let\glsifplural\@firstoftwo
1557 ]%
1558 {#1}{#2}%
1559 {\@gls@field@font{\Glsaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
1560 }

```

\@glssymbol@

```

1561 \def\@glssymbol@#1#2[#3]{%
1562 \glstrassignfieldfont{#2}%
1563 \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
1564 }

```

\@GLssymbol@ First letter uppercase version.

```

1565 \def\@GLssymbol@#1#2[#3]{%
1566 \glstrassignfieldfont{#2}%
1567 \@gls@field@link
1568 [\let\glscapscase\@secondoftwo]%
1569 {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
1570 }

```

\@GLSsymbol@ All uppercase version.

```

1571 \def\@GLSsymbol@#1#2[#3]{%
1572 \glstrassignfieldfont{#2}%
1573 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1574 {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
1575 }

```

lssymbolplural@ No case-changing version.

```

1576 \def\@glssymbolplural@#1#2[#3]{%
1577 \glstrassignfieldfont{#2}%
1578 \@gls@field@link
1579 [\let\glscapscase\@secondoftwo
1580 \let\glsifplural\@firstoftwo
1581 ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
1582 }

```

lssymbolplural@ First letter uppercase version.

```
1583 \def\@Glsymbolplural@#1#2[#3]{%
1584   \glstrassignfieldfont{#2}%
1585   \@gls@field@link
1586   [\let\glscapscase\@secondoftwo
1587    \let\glsifplural\@firstoftwo
1588   ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
1589 }
```

LSsymbolplural@ All uppercase version.

```
1590 \def\@GLSymbol@#1#2[#3]{%
1591   \glstrassignfieldfont{#2}%
1592   \@gls@field@link
1593   [\let\glscapscase\@thirdoftwo
1594    \let\glsifplural\@firstoftwo
1595   ]%
1596   {#1}{#2}%
1597   {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
1598 }
```

\@Glsuseri@ First letter uppercase version.

```
1599 \def\@Glsuseri@#1#2[#3]{%
1600   \glstrassignfieldfont{#2}%
1601   \@gls@field@link
1602   [\let\glscapscase\@secondoftwo]{#1}{#2}%
1603   {\@gls@field@font{\Glsentryuseri{#2}#3}}%
1604 }
```

\@GLSuseri@ All uppercase version.

```
1605 \def\@GLSuseri@#1#2[#3]{%
1606   \glstrassignfieldfont{#2}%
1607   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1608   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}}%
1609 }
```

\@Glsuserii@ First letter uppercase version.

```
1610 \def\@Glsuserii@#1#2[#3]{%
1611   \glstrassignfieldfont{#2}%
1612   \@gls@field@link
1613   [\let\glscapscase\@secondoftwo]%
1614   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
1615 }
```

\@GLSuserii@ All uppercase version.

```
1616 \def\@GLSuserii@#1#2[#3]{%
1617   \glstrassignfieldfont{#2}%
1618   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1619   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}}%
1620 }
```

\@Glsuseriii@ First letter uppercase version.

```
1621 \def\@Glsuseriii@#1#2[#3]{%
1622   \glstrassignfieldfont{#2}%
1623   \@gls@field@link
1624   [\let\glscapscase\@secondoftwo]%
1625   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
1626 }
```

\@GLSuseriii@ All uppercase version.

```
1627 \def\@GLSuseriii@#1#2[#3]{%
1628   \glstrassignfieldfont{#2}%
1629   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1630   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriii{#2}#3}}}%
1631 }
```

\@Glsuseriv@ First letter uppercase version.

```
1632 \def\@Glsuseriv@#1#2[#3]{%
1633   \glstrassignfieldfont{#2}%
1634   \@gls@field@link
1635   [\let\glscapscase\@secondoftwo]%
1636   {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
1637 }
```

\@GLSuseriv@ All uppercase version.

```
1638 \def\@GLSuseriv@#1#2[#3]{%
1639   \glstrassignfieldfont{#2}%
1640   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1641   {#1}{#2}%
1642   {\@gls@field@font{\mfirstucMakeUppercase{\glentryuseriv{#2}#3}}}%
1643 }
```

\@Glsuserv@ First letter uppercase version.

```
1644 \def\@Glsuserv@#1#2[#3]{%
1645   \glstrassignfieldfont{#2}%
1646   \@gls@field@link
1647   [\let\glscapscase\@secondoftwo]%
1648   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
1649 }
```

\@GLSuserv@ All uppercase version.

```
1650 \def\@GLSuserv@#1#2[#3]{%
1651   \glstrassignfieldfont{#2}%
1652   \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1653   {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuserv{#2}#3}}}%
1654 }
```

\@Glsuservi@ First letter uppercase version.

```
1655 \def\@Glsuservi@#1#2[#3]{%
```

```

1656 \glstrassignfieldfont{#2}%
1657 \@gls@field@link
1658 [\let\glscapscase\@secondoftwo]%
1659 {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}}%
1660 }

```

\@GLSuservi@ All uppercase version.

```

1661 \def\@GLSuservi@#1#2[#3]{%
1662 \glstrassignfieldfont{#2}%
1663 \@gls@field@link[\let\glscapscase\@thirdoftwo]%
1664 {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glentryuservi{#2}#3}}}%
1665 }

```

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

\@acrshort No case change.

```

1666 \def\@acrshort#1#2[#3]{%
1667 \glsoifexists{#2}%
1668 {%
1669 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1670 \let\glstrifwasfirstuse\@secondoftwo
1671 \let\glsifplural\@secondoftwo
1672 \let\glscapscase\@firstofthree
1673 \let\glsinsert\@empty
1674 \def\glscustomtext{%
1675 \acronymfont{\Glsaccesssshort{#2}}#3%
1676 }%
1677 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1678 }%
1679 \glspostlinkhook
1680 }

```

\@Acrshort First letter uppercase.

```

1681 \def\@Acrshort#1#2[#3]{%
1682 \glsoifexists{#2}%
1683 {%
1684 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1685 \let\glstrifwasfirstuse\@secondoftwo
1686 \let\glsifplural\@secondoftwo
1687 \let\glscapscase\@secondofthree
1688 \let\glsinsert\@empty
1689 \def\glscustomtext{%
1690 \Glsaccesssshort{#2}}#3%
1691 }%
1692 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1693 }%
1694 \glspostlinkhook
1695 }

```

\@ACRshort All uppercase.

```
1696 \def\@ACRshort#1#2[#3]{%
1697   \glsdoifexists{#2}%
1698   {%
1699     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1700     \let\glxtrifwasfirstuse\@secondoftwo
1701     \let\gl@sifplural\@secondoftwo
1702     \let\glscapscase\@thirdofthree
1703     \let\gl$insert\@empty
1704     \def\glscustomtext{%
1705       \mfirstucMakeUppercase{\acronymfont{\gl@saccesssshort{#2}}#3}%
1706     }%
1707     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1708   }%
1709   \glspostlinkhook
1710 }
```

\@acrshortpl No case change.

```
1711 \def\@acrshortpl#1#2[#3]{%
1712   \glsdoifexists{#2}%
1713   {%
1714     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1715     \let\glxtrifwasfirstuse\@secondoftwo
1716     \let\gl@sifplural\@firstoftwo
1717     \let\glscapscase\@firstofthree
1718     \let\gl$insert\@empty
1719     \def\glscustomtext{%
1720       \acronymfont{\gl@saccesssshortpl{#2}}#3%
1721     }%
1722     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1723   }%
1724   \glspostlinkhook
1725 }
```

\@Acrshortpl First letter uppercase.

```
1726 \def\@Acrshortpl#1#2[#3]{%
1727   \glsdoifexists{#2}%
1728   {%
1729     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1730     \let\glxtrifwasfirstuse\@secondoftwo
1731     \let\gl@sifplural\@firstoftwo
1732     \let\glscapscase\@secondofthree
1733     \let\gl$insert\@empty
1734     \def\glscustomtext{%
1735       \acronymfont{\Glsaccesssshortpl{#2}}#3%
1736     }%
1737     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1738   }%
1739   \glspostlinkhook
```

1740 }

\@ACRshortpl All uppercase.

```
1741 \def\@ACRshortpl#1#2[#3]{%
1742   \glsdoifexists{#2}%
1743   {%
1744     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1745     \let\glxtrifwasfirstuse\@secondoftwo
1746     \let\glsifplural\@firstoftwo
1747     \let\glscapscase\@thirdofthree
1748     \let\glsinsert\@empty
1749     \def\glscustomtext{%
1750       \mfirstucMakeUppercase{\acronymfont{\glsaccessshortpl{#2}}#3}%
1751     }%
1752     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1753   }%
1754   \glspostlinkhook
1755 }
```

\@acrlong No case change.

```
1756 \def\@acrlong#1#2[#3]{%
1757   \glsdoifexists{#2}%
1758   {%
1759     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1760     \let\glxtrifwasfirstuse\@secondoftwo
1761     \let\glsifplural\@secondoftwo
1762     \let\glscapscase\@firstofthree
1763     \let\glsinsert\@empty
1764     \def\glscustomtext{%
1765       \acronymfont{\glsaccesslong{#2}}#3%
1766     }%
1767     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1768   }%
1769   \glspostlinkhook
1770 }
```

\@Acrlong First letter uppercase.

```
1771 \def\@Acrlong#1#2[#3]{%
1772   \glsdoifexists{#2}%
1773   {%
1774     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1775     \let\glxtrifwasfirstuse\@secondoftwo
1776     \let\glsifplural\@secondoftwo
1777     \let\glscapscase\@secondofthree
1778     \let\glsinsert\@empty
1779     \def\glscustomtext{%
1780       \acronymfont{\Glsaccesslong{#2}}#3%
1781     }%
1782     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%

```

```

1783 }%
1784 \glspostlinkhook
1785 }

```

\@ACRlong All uppercase.

```

1786 \def\@ACRlong#1#2[#3]{%
1787 \glsdoidexists{#2}%
1788 {%
1789 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1790 \let\glxtrifwasfirstuse\@secondoftwo
1791 \let\gl@sifplural\@secondoftwo
1792 \let\glscapscase\@thirdofthree
1793 \let\gl$insert\@empty
1794 \def\glscustomtext{%
1795 \mfirstucMakeUppercase{\acronymfont{\gl@saccesslong{#2}}#3}%
1796 }%
1797 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1798 }%
1799 \glspostlinkhook
1800 }

```

\@acrlongpl No case change.

```

1801 \def\@acrlongpl#1#2[#3]{%
1802 \glsdoidexists{#2}%
1803 {%
1804 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1805 \let\glxtrifwasfirstuse\@secondoftwo
1806 \let\gl@sifplural\@firstoftwo
1807 \let\glscapscase\@firstofthree
1808 \let\gl$insert\@empty
1809 \def\glscustomtext{%
1810 \acronymfont{\gl@saccesslongpl{#2}}#3%
1811 }%
1812 \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1813 }%
1814 \glspostlinkhook
1815 }

```

\@Acrlongpl First letter uppercase.

```

1816 \def\@Acrlongpl#1#2[#3]{%
1817 \glsdoidexists{#2}%
1818 {%
1819 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
1820 \let\glxtrifwasfirstuse\@secondoftwo
1821 \let\gl@sifplural\@firstoftwo
1822 \let\glscapscase\@secondofthree
1823 \let\gl$insert\@empty
1824 \def\glscustomtext{%
1825 \acronymfont{\Glsaccesslongpl{#2}}#3%

```

```

1826 }%
1827 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1828 }%
1829 \glspostlinkhook
1830 }

```

\@ACRlongpl All uppercase.

```

1831 \def\@ACRlongpl#1#2[#3]{%
1832 \glsdoifexists{#2}%
1833 {%
1834 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
1835 \let\glstrifwasfirstuse\@secondoftwo
1836 \let\glsifplural\@firstoftwo
1837 \let\glscapscase\@thirdofthree
1838 \let\glsinsert\@empty
1839 \def\glscustomtext{%
1840 \mfirstucMakeUppercase{\acronymfont{\glsaccesslongpl{#2}}#3}%
1841 }%
1842 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
1843 }%
1844 \glspostlinkhook
1845 }

```

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

\glsaddkey

```

1846 \renewcommand*{\@glsaddkey}[7]{%
1847 \key@ifundefined{glossentry}{#1}%
1848 {%
1849 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
1850 \appto\@gls@keymap{,{#1}{#1}}%
1851 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
1852 \appto\@newglossaryentryposthook{%
1853 \letcs{\@glo@tmp}{@glo@#1}%
1854 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
1855 }%
1856 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
1857 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%

```

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

```

1858 \ifcsdef{@gls@user@#1@}%
1859 {%
1860 \PackageError{glossaries}%
1861 {Can't define '\string#5' as helper command
1862 '\expandafter\string\csname @gls@user@#1@\endcsname' already
1863 exists}%
1864 }%
1865 }%
1866 {%

```



```

1867 \expandafter\newcommand\expandafter*\expandafter
1868 {\csname @gls@user@#1\endcsname}[2][\%
1869 \new@ifnextchar[
1870 {\csuse{@gls@user@#1@}{##1}{##2}}%
1871 {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
1872 \csdef{@gls@user@#1@}##1##2[##3]{%
1873 \@gls@field@link{##1}{##2}{#3{##2}##3}%
1874 }%
1875 \newrobustcmd*{#5}{%
1876 \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
1877 }%

```

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

```

1878 \ifcsdef{@Gls@user@#1@}%
1879 {%
1880 \PackageError{glossaries}%
1881 {Can't define '\string#6' as helper command
1882 '\expandafter\string\csname @Gls@user@#1\endcsname' already
1883 exists}%
1884 }%
1885 }%
1886 {%
1887 \expandafter\newcommand\expandafter*\expandafter
1888 {\csname @Gls@user@#1\endcsname}[2][\%
1889 \new@ifnextchar[
1890 {\csuse{@Gls@user@#1@}{##1}{##2}}%
1891 {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
1892 \csdef{@Gls@user@#1@}##1##2[##3]{%
1893 \@gls@field@link[\let\glscaps@case\@secondofthree]%
1894 {##1}{##2}{#4{##2}##3}%
1895 }%
1896 \newrobustcmd*{#6}{%
1897 \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
1898 }%

```

Finally the all caps version (modified):

```

1899 \ifcsdef{@GLS@user@#1@}%
1900 {%
1901 \PackageError{glossaries}%
1902 {Can't define '\string#7' as helper command
1903 '\expandafter\string\csname @GLS@user@#1\endcsname' already
1904 exists}%
1905 }%
1906 }%
1907 {%
1908 \expandafter\newcommand\expandafter*\expandafter
1909 {\csname @GLS@user@#1\endcsname}[2][\%
1910 \new@ifnextchar[
1911 {\csuse{@GLS@user@#1@}{##1}{##2}}%
1912 {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%

```

```

1913 \csdef{@GLS@user@#1@}##1##2[##3]{%
1914 \@gls@field@link[\let\glscaps@case\@thirdofthree]%
1915 {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}}%
1916 }%
1917 \newrobustcmd*{#7}{%
1918 \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
1919 }%
1920 }%
1921 {%
1922 \PackageError{glossaries-extra}{Key ‘#1’ already exists}{}%
1923 }%
1924 }

```

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

```

1925 \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.

```

1926 \let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
1927 \renewcommand*{\@gls@link@checkfirsthyper}{%

```

\ifglused 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.

```

1928 \ifglused{\glslabel}%
1929 {\let\glsxtrifwasfirstuse\@secondoftwo}
1930 {\let\glsxtrifwasfirstuse\@firstoftwo}%

```

Store the category label for convenience.

```

1931 \edef\glscategorylabel{\glscategory{\glslabel}}%
1932 \ifglused{\glslabel}%
1933 {%
1934 \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
1935 {\KV@glslink@hyperfalse}}}%
1936 }%
1937 {%
1938 \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
1939 {\KV@glslink@hyperfalse}}}%
1940 }%
1941 \glslinkcheckfirsthyperhook
1942 }

```

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.

```

1943 \ifdef\do@glstdisablehyperinlist
1944 {%
1945 \let\@glsxtr@do@glstdisablehyperinlist\do@glstdisablehyperinlist
1946 \renewcommand*{\do@glstdisablehyperinlist}{%

```

```

1947 \glstr@do@gl:disablehyperinlist
1948 \gl@ifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}%
1949 }
1950 }
1951 {}

```

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

```

1952 \define@boolkey{glslink}{noindex}[true]{}
1953 \KV@glslink@noindexfalse

```

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

`lt@glslink@opts`

```

1954 \ifdef\@gl@setdefault@glslink@opts
1955 {
1956 \renewcommand*{\@gl@setdefault@glslink@opts}{%
1957 \KV@glslink@noindexfalse
1958 \@glstrsetaliasnoindex
1959 }
1960 }
1961 {

```

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

```

1962 \newcommand*{\@gl@setdefault@glslink@opts}{%
1963 \KV@glslink@noindexfalse
1964 \@glstrsetaliasnoindex
1965 }
1966 \preto\do@gl:disablehyperinlist{\@gl@setdefault@glslink@opts}
1967 }

```

`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.)

```

1968 \providecommand*{\glstrsetaliasnoindex}{%
1969 \KV@glslink@noindextrue
1970 }

```

`setaliasnoindex`

```

1971 \newcommand*{\@glstrsetaliasnoindex}{%
1972 \ifglshasfield{alias}{\glslabel}%
1973 {%
1974 \let\glstrindexaliased\@glstrindexaliased
1975 \glstrsetaliasnoindex
1976 \let\glstrindexaliased\@no@glstrindexaliased
1977 }%
1978 {}%
1979 }

```

xtrindexaliased

```
1980 \newcommand{\@glxtrindexaliased}{%
1981   \ifKV@glslink@noindex
1982   \else
1983     \begingroup
1984     \def\@glslnumberformat{glslnumberformat}%
1985     \edef\@glsl@counter{\csname glo@glslsetoklabel{\glslabel}@counter\endcsname}%
1986     \glxtr@saveentrycounter
1987     \@do@wrglossary{\glxtralias{\glslabel}}%
1988     \endgroup
1989   \fi
1990 }
```

xtrindexaliased

```
1991 \newcommand{\@no@glxtrindexaliased}{%
1992   \PackageError{glossaries-extra}{\string\glxtrindexaliased\space
1993     not permitted outside definition of \string\glxtrsetaliasnoindex}%
1994   {}%
1995 }
```

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

```
1996 \let\glxtrindexaliased\@no@glxtrindexaliased
```

tDefaultGlsOpts Set the default options for \glslink etc.

```
1997 \newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
1998   \renewcommand*{\@glslsetdefault@glslink@opts}{%
1999     \setkeys{glslink}{#1}%
2000     \@glxtrsetaliasnoindex
2001   }%
2002 }
```

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

```
2003 \newcommand*{\glxtrifindexing}[2]{%
2004   \ifKV@glslink@noindex #2\else #1\fi
2005 }
```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```
2006 \renewcommand*{\glswriteentry}[2]{%
2007   \glxtrifindexing
2008   {%
2009     \ifglslindexonlyfirst
2010       \ifglslused{#1}
2011       {\glxtrdoautoindexname{#1}{dualindex}}%
2012       {#2}%
2013     \else
2014       \glslifattribute{#1}{indexonlyfirst}{true}%
2015       {\ifglslused{#1}
2016         {\glxtrdoautoindexname{#1}{dualindex}}%

```

```

2017      {\#2}}%
2018      {\#2}%
2019      \fi
2020    }%
2021    {}%
2022 }

```

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

```

2023 \appto@@do@@wrglossary{\@glxtr@do@@wrindex
2024   \glxtrdowrglossaryhook{\@gls@label}}%
2025 }

```

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

Similarly for the “noidx” version:

`s@noidxglossary`

```

2026 \appto\gls@noidxglossary{\@glxtr@do@@wrindex
2027   \glxtrdowrglossaryhook{\@gls@label}}%
2028 }

```

`xtr@do@@wrindex`

```

2029 \newcommand*{\@glxtr@do@@wrindex}{%
2030   \glxtrdoautoindexname{\@gls@label}{dualindex}}%
2031 }

```

`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.)

```

2032 \newcommand*{\glxtrdowrglossaryhook}[1]{%

```

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

```

2033 \newcommand*{\@gls@alt@hyp@opt}[1]{%
2034   \let\glslinkvar\@firstofthree
2035   \let\@gls@hyp@opt@cs#1\relax
2036   \@ifstar{\s@gls@hyp@opt}%
2037   {\@ifnextchar+%
2038     {\@firstoftwo{\p@gls@hyp@opt}}%
2039     {%
2040       \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
2041       {\@firstoftwo{\@alt@gls@hyp@opt}}%
2042       {\#1}%
2043     }}%
2044 }%
2045 }

```

`alt@gls@hyp@opt` User version

```

2046 \newcommand*{\@alt@gls@hyp@opt}[1][ ]{%

```

```

2047 \let\glslinkvar\@firstofthree
2048 \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}

```

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

```

2049 \newcommand*{\@gls@alt@hyp@opt@char}{}

```

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

```

2050 \newcommand*{\@gls@alt@hyp@opt@keys}{}

```

rSetAltModifier

```

2051 \newcommand*{\GlsXtrSetAltModifier}[2]{%
2052   \let\@gls@hyp@opt\@gls@alt@hyp@opt
2053   \def\@gls@alt@hyp@opt@char{#1}%
2054   \def\@gls@alt@hyp@opt@keys{#2}%
2055 }

```

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

```

2056 \renewcommand*{\glsdohyperlink}[2]{%
2057   \glsattribute{\glslabel}{targeturl}%
2058   {%
2059     \glsattribute{\glslabel}{targetname}%
2060     {%
2061       \glsattribute{\glslabel}{targetcategory}%
2062       {%
2063         \hyperref{\glsattribute{\glslabel}{targeturl}}%
2064         {\glsattribute{\glslabel}{targetcategory}}%
2065         {\glsattribute{\glslabel}{targetname}}%
2066         {\glsxtrprotectlinks#2}}%
2067       }%
2068     }%
2069     \hyperref{\glsattribute{\glslabel}{targeturl}}%
2070     {%
2071       {\glsattribute{\glslabel}{targetname}}%
2072       {\glsxtrprotectlinks#2}}%
2073     }%
2074   }%
2075   {%
2076     \href{\glsattribute{\glslabel}{targeturl}}%
2077     {\glsxtrprotectlinks#2}}%
2078   }%
2079 }%
2080 {%

```

Check for alias.

```
2081 \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
2082 \ifdefvoid\gloaliaslabel
2083 {%
2084   \hyperlink{#1}{\glstrprotectlinks#2}}%
2085 }%
2086 {%
```

Redirect link to the alias target.

```
2087   \hyperlink
2088   {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
2089   {\glstrprotectlinks#2}}%
2090 }%
2091 }%
2092 }
```

`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.

```
2093 \renewrobustcmd*{\glshyperlink}[2][\glstrytext{\@glo@label}]{%
2094   \def\@glo@label{#2}%
2095   {\edef\glslabel{#2}%
2096     \@glslink{\glolinkprefix\glslabel}{#1}}%
2097 }
```

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

```
2098 \ifundef\glsdohyperlink
2099 {%
2100   \renewcommand{\glsdisablehyper}{%
2101     \KV@glslink@hyperfalse
2102     \let\@glslink\glsdohyperlink
2103     \let\@glstarget\@secondoftwo
2104   }
2105 }
2106 {}
```

`glsdohyperlink` 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.

```
2107 \def\glsdohyperlink#1#2{\glstrprotectlinks #2}
```

Reset `\@glslink` with patched versions:

```
2108 \ifcsundef{hyperlink}%
2109 {%
2110   \let\@glslink\glsdohyperlink
2111 }%
2112 {%
```

```

2113 \let\@glslink\glsdohyperlink
2114 }

```

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

```

2115 \newcommand*{\glsxtrprotectlinks}{%
2116   \KV@glslink@hyperfalse
2117   \KV@glslink@noindextrue
2118   \let\@gls\@glsxtr@p@text@
2119   \let\@Gls\@Glsxtr@p@text@
2120   \let\@GLS\@GLSxtr@p@text@
2121   \let\@glspl\@glsxtr@p@plural@
2122   \let\@Glspl\@Glsxtr@p@plural@
2123   \let\@GLSpl\@GLSxtr@p@plural@
2124   \let\@glsxtrshort\@glsxtr@p@short@
2125   \let\@Glsxtrshort\@Glsxtr@p@short@
2126   \let\@GLSxtrshort\@GLSxtr@p@short@
2127   \let\@glsxtrlong\@glsxtr@p@long@
2128   \let\@Glsxtrlong\@Glsxtr@p@long@
2129   \let\@GLSxtrlong\@GLSxtr@p@long@
2130   \let\@glsxtrshortpl\@glsxtr@p@shortpl@
2131   \let\@Glsxtrshortpl\@Glsxtr@p@shortpl@
2132   \let\@GLSxtrshortpl\@GLSxtr@p@shortpl@
2133   \let\@glsxtrlongpl\@glsxtr@p@longpl@
2134   \let\@Glsxtrlongpl\@Glsxtr@p@longpl@
2135   \let\@GLSxtrlongpl\@GLSxtr@p@longpl@
2136   \let\@acrshort\@glsxtr@p@acrshort@
2137   \let\@Acrshort\@Glsxtr@p@acrshort@
2138   \let\@ACRshort\@GLSxtr@p@acrshort@
2139   \let\@acrshortpl\@glsxtr@p@acrshortpl@
2140   \let\@Acrshortpl\@Glsxtr@p@acrshortpl@
2141   \let\@ACRshortpl\@GLSxtr@p@acrshortpl@
2142   \let\@acrlong\@glsxtr@p@acrlong@
2143   \let\@Acrlong\@Glsxtr@p@acrlong@
2144   \let\@ACRlong\@GLSxtr@p@acrlong@
2145   \let\@acrlongpl\@glsxtr@p@acrlongpl@
2146   \let\@Acrlongpl\@Glsxtr@p@acrlongpl@
2147   \let\@ACRlongpl\@GLSxtr@p@acrlongpl@
2148 }

```

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

`@glsxtr@p@text@`

```

2149 \def\@glsxtr@p@text@#1#2[#3]{\@glstext@{#1}{#2}[#3]}

```

`@Glsxtr@p@text@`

```

2150 \def\@Glsxtr@p@text@#1#2[#3]{\@Glstext@{#1}{#2}[#3]}

```

`@GLSxtr@p@text@`

```

2151 \def\@GLSxtr@p@text@#1#2[#3]{\@GLStext@{#1}{#2}[#3]}

```


lsxtr@p@plural@

```
2152 \def\@glxtr@p@plural@#1#2[#3]{\@glsplural@{#1}{#2}[#3]}
```

lsxtr@p@plural@

```
2153 \def\@Glsxtr@p@plural@#1#2[#3]{\@Glsplural@{#1}{#2}[#3]}
```

LSxtr@p@plural@

```
2154 \def\@GLSxtr@p@plural@#1#2[#3]{\@GLSplural@{#1}{#2}[#3]}
```

glxtr@p@short@

```
2155 \def\@glxtr@p@short@#1#2[#3]{%
2156   {%
2157     \glsetabbrvfmt{\glscategory{#2}}%
2158     \glabbrvfont{\glentryshort{#2}}#3%
2159   }%
2160 }
```

Glsxtr@p@short@

```
2161 \def\@Glsxtr@p@short@#1#2[#3]{%
2162   {%
2163     \glsetabbrvfmt{\glscategory{#2}}%
2164     \glabbrvfont{\Glsentryshort{#2}}#3%
2165   }%
2166 }
```

GLSxtr@p@short@

```
2167 \def\@GLSxtr@p@short@#1#2[#3]{%
2168   {%
2169     \glsetabbrvfmt{\glscategory{#2}}%
2170     \mfirstucMakeUppercase{\glabbrvfont{\glentryshort{#2}}#3}%
2171   }%
2172 }
```

sxtr@p@shortpl@

```
2173 \def\@glxtr@p@shortpl@#1#2[#3]{%
2174   {%
2175     \glsetabbrvfmt{\glscategory{#2}}%
2176     \glabbrvfont{\glentryshortpl{#2}}#3%
2177   }%
2178 }
```

sxtr@p@shortpl@

```
2179 \def\@Glsxtr@p@shortpl@#1#2[#3]{%
2180   {%
2181     \glsetabbrvfmt{\glscategory{#2}}%
2182     \glabbrvfont{\Glsentryshortpl{#2}}#3%
2183   }%
2184 }
```

Sxtr@p@shortpl@

```

2185 \def\@GLSxtr@p@shortpl@#1#2[#3]{%
2186   {%
2187     \glsetabbrvfmt{\glscategory{#2}}%
2188     \mfirstucMakeUppercase{\glsabbrvfont{\glentryshortpl{#2}}#3}%
2189   }%
2190 }

```

@glxtr@p@long@

```

2191 \def\@glxtr@p@long@#1#2[#3]{\glentrylong{#2}#3}

```

@Glsxtr@p@long@

```

2192 \def\@Glsxtr@p@long@#1#2[#3]{\Glsentrylong{#2}#3}

```

@GLSxtr@p@long@

```

2193 \def\@GLSxtr@p@long@#1#2[#3]{%
2194   {\mfirstucMakeUppercase{\glslongfont{\glentrylong{#2}}#3}}

```

lsxtr@p@longpl@

```

2195 \def\@glxtr@p@longpl@#1#2[#3]{\glentrylongpl{#2}#3}

```

lSxtr@p@longpl@

```

2196 \def\@Glsxtr@p@longpl@#1#2[#3]{\glslongfont{\Glsentrylongpl{#2}}#3}

```

LSxtr@p@longpl@

```

2197 \def\@GLSxtr@p@longpl@#1#2[#3]{%
2198   {\mfirstucMakeUppercase{\glslongfont{\glentrylongpl{#2}}#3}}

```

xtr@p@acrshort@

```

2199 \def\@glxtr@p@acrshort@#1#2[#3]{\acronymfont{\glentryshort{#2}}#3}

```

xtr@p@acrshort@

```

2200 \def\@Glsxtr@p@acrshort@#1#2[#3]{\acronymfont{\Glsentryshort{#2}}#3}

```

xtr@p@acrshort@

```

2201 \def\@GLSxtr@p@acrshort@#1#2[#3]{%
2202   {\mfirstucMakeUppercase{\acronymfont{\glentryshort{#2}}#3}}

```

r@p@acrshortpl@

```

2203 \def\@glxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\glentryshortpl{#2}}#3}

```

r@p@acrshortpl@

```

2204 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{\acronymfont{\Glsentryshortpl{#2}}#3}

```

r@p@acrshortpl@

```

2205 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
2206   {\mfirstucMakeUppercase{\acronymfont{\glentryshortpl{#2}}#3}}

```

```

sctr@p@acrlong@
2207 \def\@glxstr@p@acrlong@#1#2[#3]{\@glstrylong{#2}#3}}

sctr@p@acrlong@
2208 \def\@Glsxtr@p@acrlong@#1#2[#3]{\@Glsstrylong{#2}#3}}

Sxtr@p@acrlong@
2209 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
2210 {\mfirstucMakeUppercase{\@glstrylong{#2}#3}}}}

tr@p@acrlongpl@
2211 \def\@glxstr@p@acrlongpl@#1#2[#3]{\@glstrylongpl{#2}#3}}

tr@p@acrlongpl@
2212 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{\@Glsstrylongpl{#2}#3}}

tr@p@acrlongpl@
2213 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
2214 {\mfirstucMakeUppercase{\@glstrylongpl{#2}#3}}}}

```

Commands to minimise conflict.

```

\@glxstrp@opt
2215 \newcommand*{\@glxstrp@opt}{hyper=false,noindex}

\glxstrsetpopts  Used in glossary to switch hyperlinks on for the \@glxstrp type of commands.
2216 \newcommand*{\glxstrsetpopts}[1]{%
2217   \renewcommand*{\@glxstrp@opt}{#1}%
2218 }

lossxtrsetpopts  Used in glossary to switch hyperlinks on for the \@glxstrp type of commands.
2219 \newcommand*{\glossxtrsetpopts}{%
2220   \glxstrsetpopts{noindex}%
2221 }

\@@glxstrp
2222 \newrobustcmd*{\@@glxstrp}[2]{%
  Add scope.
2223   {%
2224     \let\glspostlinkhook\relax
2225     \csname#1\expandafter\endcsname\expandafter[\@glxstrp@opt]{#2}[]%
2226   }%
2227 }

```

\@glsxtrp

```
2228 \newrobustcmd*{\@glsxtrp}[2]{%
2229   \ifcsdef{gls#1}%
2230   {%
2231     \@glsxtrp{gls#1}{#2}%
2232   }%
2233   {%
2234     \ifcsdef{glsxtr#1}%
2235     {%
2236       \@glsxtrp{glsxtr#1}{#2}%
2237     }%
2238     {%
2239       \PackageError{glossaries-extra}{‘#1’ not recognised by
2240       \string\glsxtrp}{}%
2241     }%
2242   }%
2243 }
```

\@Glsxtrp

```
2244 \newrobustcmd*{\@Glsxtrp}[2]{%
2245   \ifcsdef{Gls#1}%
2246   {%
2247     \@glsxtrp{Gls#1}{#2}%
2248   }%
2249   {%
2250     \ifcsdef{Glsxtr#1}%
2251     {%
2252       \@glsxtrp{Glsxtr#1}{#2}%
2253     }%
2254     {%
2255       \PackageError{glossaries-extra}{‘#1’ not recognised by
2256       \string\Glsxtrp}{}%
2257     }%
2258   }%
2259 }
```

\@GLSxtrp

```
2260 \newrobustcmd*{\@GLSxtrp}[2]{%
2261   \ifcsdef{GLS#1}%
2262   {%
2263     \@glsxtrp{GLS#1}{#2}%
2264   }%
2265   {%
2266     \ifcsdef{GLSxtr#1}%
2267     {%
2268       \@glsxtrp{GLSxtr#1}{#2}%
2269     }%
2270     {%
2271       \PackageError{glossaries-extra}{‘#1’ not recognised by
```

```

2272      \string\GLSxtrp}{}%
2273    }%
2274  }%
2275 }

```

\glsxtr@entry@p

```

2276 \newrobustcmd*{\glsxtr@headentry@p}[2]{%
2277   \glsifattribute{#1}{headuc}{true}%
2278   {%
2279     \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
2280   }%
2281   {%
2282     \@gls@entry@field{#1}{#2}%
2283   }%
2284 }

```

\glsxtrp Not robust as it needs to expand somewhat.

```

2285 \ifdef\teorpdfstring
2286 {
2287   \newcommand{\glsxtrp}[2]{%
2288     \protect\NoCaseChange
2289     {%
2290       \protect\teorpdfstring
2291       {%
2292         \protect\glsxtrifinmark
2293         {%
2294           \ifcsdef{glsxtrhead#1}%
2295             {%
2296               {\protect\csuse{glsxtrhead#1}{#2}}%
2297             }%
2298             {%
2299               \glsxtr@headentry@p{#2}{#1}%
2300             }%
2301           }%
2302           {%
2303             \@glsxtrp{#1}{#2}%
2304           }%
2305         }%
2306         {%
2307           \protect\@gls@entry@field{#2}{#1}%
2308         }%
2309       }%
2310     }
2311 }
2312 {
2313   \newcommand{\glsxtrp}[2]{%
2314     \protect\NoCaseChange
2315     {%
2316       \protect\glsxtrifinmark

```

```

2317     {%
2318     \ifcsdef{glxtrhead#1}%
2319     {%
2320     {\protect\csuse{glxtrhead#1}}%
2321     }%
2322     {%
2323     \glxtr@headentry@p{#2}{#1}%
2324     }%
2325     }%
2326     {%
2327     \@glxtrp{#1}{#2}%
2328     }%
2329     }%
2330 }
2331 }

```

Provide short synonyms for the most common option.

`\glsp`

```
2332 \newcommand*{\glsp}{\glxtrp{short}}
```

`\glsp`

```
2333 \newcommand*{\glsp}{\glxtrp{text}}
```

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

```

2334 \ifdef\teorpdfstring
2335 {
2336   \newcommand{\Glsxtrp}[2]{%
2337     \protect\NoCaseChange
2338     {%
2339       \protect\teorpdfstring
2340       {%
2341         \protect\glxtrifinmark
2342         {%
2343           \ifcsdef{Glsxtrhead#1}%
2344           {%
2345             {\protect\csuse{Glsxtrhead#1}{#2}}%
2346             }%
2347             {%
2348               \protect\@Gls@entry@field{#2}{#1}%
2349               }%
2350             }%
2351             {%
2352               \@Glsxtrp{#1}{#2}%
2353               }%
2354             }%
2355             {%
2356               \protect\@Gls@entry@field{#2}{#1}%

```

```

2357     }%
2358 }%
2359 }
2360 }
2361 {
2362   \newcommand{\Glsxtrp}[2]{%
2363     \protect\NoCaseChange
2364     {%
2365       \protect\glsxtrifinmark
2366       {%
2367         \ifcsdef{Glsxtrhead#1}%
2368         {%
2369           {\protect\csuse{Glsxtrhead#1}}%
2370         }%
2371         {%
2372           \protect\@Gls@entry@field{#2}{#1}%
2373         }%
2374       }%
2375       {%
2376         \@Glsxtrp{#1}{#2}%
2377       }%
2378     }%
2379   }
2380 }

```

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

```

2381 \ifdef\textorpdfstring
2382 {
2383   \newcommand{\Glsxtrp}[2]{%
2384     \protect\NoCaseChange
2385     {%
2386       \protect\textorpdfstring
2387       {%
2388         \protect\glsxtrifinmark
2389         {%
2390           \ifcsdef{Glsxtr#1}%
2391           {%
2392             {\protect\Glsxtrshort[noindex,hyper=false]{#1}[]}%
2393           }%
2394           {%
2395             \protect\mfirstucMakeUppercase
2396             {%
2397               \protect\@gls@entry@field{#2}{#1}%
2398             }%
2399           }%
2400         }%
2401         {%
2402           \@Glsxtrp{#1}{#2}%
2403         }%

```

```

2404     }%
2405     {%
2406         \protect\@gls@entry@field{#2}{#1}%
2407     }%
2408 }%
2409 }
2410 }
2411 {
2412     \newcommand{\GLSxtrp}[2]{%
2413         \protect\NoCaseChange
2414         {%
2415             \protect\glsxtrifinmark
2416             {%
2417                 \ifcsdef{GLSxtr#1}%
2418                 {%
2419                     {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
2420                 }%
2421                 {%
2422                     \protect\mfirstucMakeUppercase
2423                     {%
2424                         \protect\@gls@entry@field{#2}{#1}%
2425                     }%
2426                 }%
2427             }%
2428         }%
2429         \@GLSxtrp{#1}{#2}%
2430     }%
2431 }%
2432 }
2433 }

```

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.

```

2434 \renewcommand*{\@glsunset}[1]{%
2435     \@glsunset{#1}%
2436     \glsxtrpostunset{#1}%
2437 }%

```

`glsxtrpostunset`

```

2438 \newcommand*{\glsxtrpostunset}[1]{%

```

`\@glslocalunset` Local unset.


```

2439 \renewcommand*{\@glslocalunset}[1]{%
2440   \@@glslocalunset{#1}%
2441   \glsxtrpostlocalunset{#1}%
2442 }%

rpostlocalunset
2443 \newcommand*{\glsxtrpostlocalunset}[1]{}

\@glsreset   Global reset.
2444 \renewcommand*{\@glsreset}[1]{%
2445   \@@glsreset{#1}%
2446   \glsxtrpostreset{#1}%
2447 }%

glsxtrpostreset
2448 \newcommand*{\glsxtrpostreset}[1]{}

\@glslocalreset   Local reset.
2449 \renewcommand*{\@glslocalreset}[1]{%
2450   \@@glslocalreset{#1}%
2451   \glsxtrpostlocalreset{#1}%
2452 }%

rpostlocalreset
2453 \newcommand*{\glsxtrpostlocalreset}[1]{}

leEntryCounting   The first argument is the list of categories and the second argument is the value of the en-
                    trycount attribute.
2454 \newcommand*{\GlsXtrEnableEntryCounting}[2]{%
    Enable entry counting:
2455   \glsenableentrycount
    Redefine \gls etc:
2456   \renewcommand*{\gls}{\cglsl}%
2457   \renewcommand*{\Gls}{\cGls}%
2458   \renewcommand*{\glspl}{\cglspl}%
2459   \renewcommand*{\Glspl}{\cGlspl}%
2460   \renewcommand*{\GLS}{\cGLS}%
2461   \renewcommand*{\GLSpl}{\cGLSpl}%
    Set the entrycount attribute:
2462   \@glsxtr@setentrycountunsetattr{#1}{#2}%
    In case this command is used again:
2463   \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
2464   \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
2465     \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
2466       can't be used with \string\GlsXtrEnableEntryCounting}%
2467     {Use one or other but not both commands}}%
2468 }

```

ycountunsetattr

```
2469 \newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
2470   \@for\@glsxtr@cat:=#1\do
2471   {%
2472     \ifdefempty{\@glsxtr@cat}{}%
2473     {%
2474       \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
2475     }%
2476   }%
2477 }
```

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

nableentrycount

```
2478 \renewcommand*{\glsenableentrycount}{%
  Enable new fields:
2479   \appto\@newglossaryentry@defcounters{\@newglossaryentry@defcounters}%
  Just in case the user has switched on the docdef option.
2480   \renewcommand*{\gls@defdocnewglossaryentry}{%
2481     \renewcommand*{\newglossaryentry}[2]{%
2482       \PackageError{glossaries}{\string\newglossaryentry\space
2483         may only be used in the preamble when entry counting has
2484         been activated}{If you use \string\glsenableentrycount\space
2485         you must place all entry definitions in the preamble not in
2486         the document environment}%
2487     }%
2488   }%
```

New commands to access new fields:

```
2489 \newcommand*{\glsentrycurrcount}[1]{%
2490   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2491   {0}{\@gls@entry@field{##1}{currcount}}%
2492 }%
2493 \newcommand*{\glsentryprevcount}[1]{%
2494   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2495   {0}{\@gls@entry@field{##1}{prevcount}}%
2496 }%
```

Adjust post unset and reset:

```
2497 \let\@glsxtr@entrycount@org@unset\glsxtrpostunset
2498 \renewcommand*{\glsxtrpostunset}[1]{%
2499   \@glsxtr@entrycount@org@unset{##1}%
2500   \@gls@increment@currcount{##1}%
2501 }%
2502 \let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
2503 \renewcommand*{\glsxtrpostlocalunset}[1]{%
2504   \@glsxtr@entrycount@org@localunset{##1}%
2505   \@gls@local@increment@currcount{##1}%
2506 }%
```

```

2507 \let\@glxtr@entrycount@org@reset\glxtrpostreset
2508 \renewcommand*{\glxtrpostreset}[1]{%
2509   \@glxtr@entrycount@org@reset{##1}%
2510   \csgdef{glo@glsdetoklabel{##1}@currcount}{0}%
2511 }%
2512 \let\@glxtr@entrycount@org@localreset\glxtrpostlocalreset
2513 \renewcommand*{\glxtrpostlocalreset}[1]{%
2514   \@glxtr@entrycount@org@localreset{##1}%
2515   \csdef{glo@glsdetoklabel{##1}@currcount}{0}%
2516 }%

```

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

```

2517 \let\@cgl@\@cgl@
2518 \let\@cgl@pl@\@cgl@pl@

2519 \let\@cGl@\@cGl@
2520 \let\@cGl@pl@\@cGl@pl@
2521 \let\@cGLS@\@cGLS@
2522 \let\@cGLSpl@\@cGLSpl@

```

The rest is as the original definition.

```

2523 \AtEndDocument{\@gl@write@entrycounts}%
2524 \renewcommand*{\@gl@entry@count}[2]{%
2525   \csgdef{glo@glsdetoklabel{##1}@prevcount}{##2}%
2526 }%
2527 \let\gl@enableentrycount\relax
2528 \renewcommand*{\gl@enableentryunitcount}{%
2529   \PackageError{glossaries-extra}{\string\gl@enableentryunitcount\space
2530     can't be used with \string\gl@enableentrycount}%
2531   {Use one or other but not both commands}%
2532 }%
2533 }

```

`@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.

```

2534 \renewcommand*{\@gl@write@entrycounts}{%
2535   \immediate\write\@auxout
2536     {\string\providecommand*{\string\@gl@entry@count}[2]{}}%
2537   \count@=0\relax
2538   \forallgl@sentries{\@gl@sentry}{%
2539     \gl@hasattribute{\@gl@sentry}{entrycount}%
2540     {%
2541       \ifgl@sused{\@gl@sentry}%
2542       {%
2543         \immediate\write\@auxout
2544           {\string\@gl@entry@count{\@gl@sentry}{\gl@sentrycurrcount{\@gl@sentry}}}%
2545       }%
2546     }%
2547   \advance\count@ by \@ne

```

```

2548 }%
2549 {}%
2550 }%
2551 \ifnum\count@=0
2552   \GlossariesExtraWarningNoLine{Entry counting has been enabled
2553   \MessageBreak with \string\glsenableentrycount\space but the
2554   \MessageBreak attribute 'entrycount' hasn't
2555   \MessageBreak been assigned to any of the defined
2556   \MessageBreak entries}%
2557 \fi
2558 }

```

trifcounttrigger

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

```

2559 \newcommand*\glxtrifcounttrigger}[3]{%
2560   \glshasattribute{#1}{entrycount}%
2561   {%
2562     \ifnum\gl Sentryprevcount{#1}>\gl sgetattribute{#1}{entrycount}\relax
2563     #3%
2564   \else
2565     #2%
2566   \fi
2567 }%
2568 {#3}%
2569 }

```

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

\@@cgl@

```

2570 \def\@@cgl@#1#2[#3]{%
2571   \glxtrifcounttrigger{#2}%
2572   {%
2573     \cgl sformat{#2}{#3}%
2574     \gl sunset{#2}%
2575   }%
2576   {%
2577     \@cgl@{#1}{#2}[#3]%
2578   }%
2579 }%

```

\@@cgl@

```

2580 \def\@@cgl spl@#1#2[#3]{%
2581   \glxtrifcounttrigger{#2}%
2582   {%
2583     \cgl splformat{#2}{#3}%
2584     \gl sunset{#2}%

```

```

2585 }%
2586 {%
2587     \@glsp1@{#1}-{#2}[#3]%
2588 }%
2589 }%

```

\@@cGls@

```

2590 \def\@@cGls@#1#2[#3]{%
2591     \glxtrifcounttrigger{#2}%
2592     {%
2593         \cGlsformat{#2}{#3}%
2594         \glset{#2}%
2595     }%
2596     {%
2597         \@Gls@{#1}-{#2}[#3]%
2598     }%
2599 }%

```

\@@cGlsp1@

```

2600 \def\@@cGlsp1@#1#2[#3]{%
2601     \glxtrifcounttrigger{#2}%
2602     {%
2603         \cGlsp1format{#2}{#3}%
2604         \glset{#2}%
2605     }%
2606     {%
2607         \@Glsp1@{#1}-{#2}[#3]%
2608     }%
2609 }%

```

\@@cGLS@

```

2610 \def\@@cGLS@#1#2[#3]{%
2611     \glxtrifcounttrigger{#2}%
2612     {%
2613         \cGLSformat{#2}{#3}%
2614         \glset{#2}%
2615     }%
2616     {%
2617         \@GLS@{#1}-{#2}[#3]%
2618     }%
2619 }%

```

\@@cGLSp1@

```

2620 \def\@@cGLSp1@#1#2[#3]{%
2621     \glxtrifcounttrigger{#2}%
2622     {%
2623         \cGLSp1format{#2}{#3}%
2624         \glset{#2}%
2625     }%

```

```

2626 {%
2627   \@GLSp1@{#1}{#2}[#3]%
2628 }%
2629 }%

```

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

```

\@cgl's@
2630 \def\@cgl's@#1#2[#3]{\@gls@{#1}{#2}[#3]}

```

```

\@cGls@
2631 \def\@cGls@#1#2[#3]{\@Gls@{#1}{#2}[#3]}

```

```

\@cgl'spl@
2632 \def\@cgl'spl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}

```

```

\@cGlspl@
2633 \def\@cGlspl@#1#2[#3]{\@Glspl@{#1}{#2}[#3]}

```

Add all upper case versions not provided by glossaries.

```

\cGLS
2634 \newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}

```

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

```

2635 \newcommand*{\@cGLS}[2][\%
2636   \new@ifnextchar[\@cGLS@{#1}{#2}]{\@cGLS@{#1}{#2}[#3]}%
2637 }

```

```

\@cGLS@
2638 \def\@cGLS@#1#2[#3]{\@GLS@{#1}{#2}[#3]}

```

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

```

2639 \newcommand*{\cGLSformat}[2]{\%
2640   \expandafter\mfirstucMakeUppercase\expandafter{\cgl'sformat{#1}{#2}}%
2641 }

```

```

\cGLSp1
2642 \newrobustcmd*{\cGLSp1}{\@gls@hyp@opt\@cGLSp1}

```

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

```

2643 \newcommand*{\@cGLSp1}[2][\%
2644   \new@ifnextchar[\@cGLSp1@{#1}{#2}]{\@cGLSp1@{#1}{#2}[#3]}%
2645 }

```

\@cGLSp1@

```
2646 \def\@cGLSp1@#1#2[#3]{\@GLSp1@{#1}{#2} [#3]}
```

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

```
2647 \newcommand*\cGLSplformat}[2]{%
```

```
2648   \expandafter\mfirstuc\MakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
```

```
2649 }
```

Modify the trigger formats to check for the regular attribute.

\cglformat

```
2650 \renewcommand*\cglformat}[2]{%
```

```
2651   \glsifregular{#1}
```

```
2652   {\glsentryfirst{#1}}%
```

```
2653   {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}\#2%
```

```
2654 }
```

\cGlsformat

```
2655 \renewcommand*\cGlsformat}[2]{%
```

```
2656   \glsifregular{#1}
```

```
2657   {\Glsentryfirst{#1}}%
```

```
2658   {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}\#2%
```

```
2659 }
```

\cglsplformat

```
2660 \renewcommand*\cglsplformat}[2]{%
```

```
2661   \glsifregular{#1}
```

```
2662   {\glsentryfirstplural{#1}}%
```

```
2663   {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}\#2%
```

```
2664 }
```

\cGlsplformat

```
2665 \renewcommand*\cGlsplformat}[2]{%
```

```
2666   \glsifregular{#1}
```

```
2667   {\Glsentryfirstplural{#1}}%
```

```
2668   {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}\#2%
```

```
2669 }
```

New code similar to above for unit counting.

defunitcounters

```
2670 \newcommand*\@newglossaryentry@defunitcounters{%
```

```
2671   \edef\@glo@countunit{\csuse{@glsxtr@categoryattr@ \@glo@category @unitcount}}%
```

```
2672   \ifdefvoid\@glo@countunit
```

```
2673   {%
```

```
2674   %
```

```
2675   \@glsxtr@ifunitcounter{\@glo@countunit}%
```

```

2676     {}%
2677     {\expandafter\@glxtr@addunitcounter\expandafter{\@glo@countunit}}%
2678 }%
2679 }

```

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

```

2680 \newcommand*{\@glxtr@unitcountlist}{}

```

`@addunitcounter`

```

2681 \newcommand*{\@glxtr@addunitcounter}[1]{%
2682   \listadd{\@glxtr@unitcountlist}{#1}%
2683   \ifcsundef{glxtr@theunit@#1}
2684   {%
2685     \ifcsdef{theH#1}%
2686     {\csdef{glxtr@theunit@#1}{\csuse{theH#1}}}%
2687     {\csdef{glxtr@theunit@#1}{\csuse{the#1}}}%
2688   }%
2689   {}%
2690 }

```

`r@ifunitcounter`

```

2691 \newcommand*{\@glxtr@ifunitcounter}[3]{%
2692   \xifinlist{#1}{\@glxtr@unitcountlist}{#2}{#3}%
2693 }

```

`urrentunitcount`

```

2694 \newcommand*\@glxtr@currentunitcount[1]{%
2695   glo@\glstoklabel{#1}@currunit@\glsggetattribute{#1}{unitcount}.%
2696   \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}%
2697 }

```

`viousunitcount`

```

2698 \newcommand*\@glxtr@previousunitcount[1]{%
2699   glo@\glstoklabel{#1}@prevunit@\glsggetattribute{#1}{unitcount}.%
2700   \csuse{glxtr@theunit@\glsggetattribute{#1}{unitcount}}%
2701 }

```

`t@currunitcount`

```

2702 \newcommand*{\@glx@increment@currunitcount}[1]{%
2703   \glshasattribute{#1}{unitcount}%
2704   {%
2705     \edef\@glxtr@csname{\@glxtr@currentunitcount{#1}}%
2706     \ifcsundef{\@glxtr@csname}%
2707     {%
2708       \csgdef{\@glxtr@csname}{1}%
2709       \listcsxadd
2710       {glo@\glstoklabel{#1}@unitlist}%
2711       {\glsggetattribute{#1}{unitcount}.%

```



```

2712         \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
2713     }%
2714 }%
2715 {%
2716     \csxdef{\@glxstr@csname}%
2717     {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
2718 }%
2719 }%
2720 {}%
2721 }

```

t@currunitcount

```

2722 \newcommand*{\@glsl@local@increment@currunitcount}[1]{%
2723     \glshasattribute{#1}{unitcount}%
2724     {%
2725         \edef\@glxstr@csname{\@glxstr@currentunitcount{#1}}%
2726         \ifcsundef{\@glxstr@csname}%
2727         {%
2728             \csdef{\@glxstr@csname}{1}%
2729             \listcseadd
2730             {glo@\glsgdetoklabel{#1}@unitlist}%
2731             {\glsggetattribute{#1}{unitcount}.%
2732             \csuse{glxstr@theunit@\glsggetattribute{#1}{unitcount}}}%
2733         }%
2734     }%
2735     {%
2736         \csedef{\@glxstr@csname}%
2737         {\number\numexpr\csname\@glxstr@csname\endcsname+1}%
2738     }%
2739 }%
2740 {}%
2741 }

```

r@currunitcount

```

2742 \newcommand*{\@glxstr@currunitcount}[2]{%
2743     \ifcsundef
2744     {glo@\glsgdetoklabel{#1}@currunit@#2}%
2745     {0}%
2746     {\csuse{glo@\glsgdetoklabel{#1}@currunit@#2}}%
2747 }%

```

r@prevunitcount

```

2748 \newcommand*{\@glxstr@prevunitcount}[2]{%
2749     \ifcsundef
2750     {glo@\glsgdetoklabel{#1}@prevunit@#2}%
2751     {0}%
2752     {\csuse{glo@\glsgdetoklabel{#1}@prevunit@#2}}%
2753 }%

```

eentryunitcount

```
2754 \newcommand*{\glsenableentryunitcount}{%
```

Enable new fields:

```
2755 \appto\@newglossaryentry@defcounters{\@newglossaryentry@defunitcounters}%
```

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

```
2756 \renewcommand*{\gls@defdocnewglossaryentry}{%
2757 \renewcommand*\newglossaryentry[2]{%
2758 \PackageError{glossaries}{\string\newglossaryentry\space
2759 may only be used in the preamble when entry counting has
2760 been activated}{If you use \string\glsenableentryunitcount\space
2761 you must place all entry definitions in the preamble not in
2762 the document environment}%
2763 }%
2764 }%
```

New commands to access new fields:

```
2765 \newcommand*{\glsentrycurrcount}[1]{%
2766 \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
2767 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
2768 }%
2769 \newcommand*{\glsentryprevcount}[1]{%
2770 \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}}.%
2771 \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
2772 }%
```

Access total count:

```
2773 \newcommand*{\glsentryprevtotalcount}[1]{%
2774 \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
2775 {0}%
2776 {%
2777 \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
2778 }%
2779 }%
```

Access max value:

```
2780 \newcommand*{\glsentryprevmaxcount}[1]{%
2781 \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
2782 {0}%
2783 {%
2784 \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
2785 }%
2786 }%
```

Adjust post unset and reset:

```
2787 \let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
2788 \renewcommand*{\glsxtrpostunset}[1]{%
2789 \@glsxtr@entryunitcount@org@unset{##1}%
2790 \@gls@increment@currunitcount{##1}%
2791 }%
2792 \let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
```

```

2793 \renewcommand*{\glxtrpostlocalunset}[1]{%
2794   \@glxtr@entryunitcount@org@localunset{##1}%
2795   \@glx@local@increment@currunitcount{##1}%
2796 }%
2797 \let\@glxtr@entryunitcount@org@reset\glxtrpostreset
2798 \renewcommand*{\glxtrpostreset}[1]{%
2799   \glshasattribute{##1}{unitcount}%
2800   {%
2801     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
2802     \ifcsundef{\@glxtr@csname}%
2803     {}%
2804     {\csgdef{\@glxtr@csname}{0}}%
2805   }%
2806   {}%
2807 }%
2808 \let\@glxtr@entryunitcount@org@localreset\glxtrpostlocalreset
2809 \renewcommand*{\glxtrpostlocalreset}[1]{%
2810   \@glxtr@entryunitcount@org@localreset{##1}%
2811   \glshasattribute{##1}{unitcount}%
2812   {%
2813     \edef\@glxtr@csname{\@glxtr@currentunitcount{##1}}%
2814     \ifcsundef{\@glxtr@csname}%
2815     {}%
2816     {\csdef{\@glxtr@csname}{0}}%
2817   }%
2818   {}%
2819 }%

```

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

```

2820 \let\@cglx@ \@cglx@
2821 \let\@cglxpl@ \@cglxpl@

2822 \let\@cGlx@ \@cGlx@
2823 \let\@cGlxpl@ \@cGlxpl@
2824 \let\@cGLS@ \@cGLS@
2825 \let\@cGLSpl@ \@cGLSpl@

```

Write information to the aux file.

```

2826 \AtEndDocument{\@glx@write@entryunitcounts}%
2827 \renewcommand*{\@glx@entry@unitcount}[3]{%
2828   \csgdef{glo@glx@detoklabel{##1}@prevunit@##3}{##2}%
2829   \ifcsundef{glo@glx@detoklabel{##1}@prevunittotal}%
2830   {\csgdef{glo@glx@detoklabel{##1}@prevunittotal}{##2}}%
2831   {%
2832     \csxdef{glo@glx@detoklabel{##1}@prevunittotal}{
2833       \number\numexpr\csuse{glo@glx@detoklabel{##1}@prevunittotal}+##2}%
2834     }%
2835     \ifcsundef{glo@glx@detoklabel{##1}@prevunitmax}%
2836     {\csgdef{glo@glx@detoklabel{##1}@prevunitmax}{##2}}%

```

```

2837   {%
2838     \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
2839       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
2840     \fi
2841   }%
2842 }%
2843 \let\glsenableentryunitcount\relax
2844 \renewcommand*{\glsenableentrycount}{%
2845   \PackageError{glossaries-extra}{\string\glsenableentrycount\space
2846     can't be used with \string\glsenableentryunitcount}%
2847   {Use one or other but not both commands}%
2848 }%
2849 }
2850 \@onlypreamble\glsenableentryunitcount

```

entry@unitcount

```

2851 \newcommand*{\@gls@entry@unitcount}[3]{}

```

ryunitcounts@do

```

2852 \newcommand*{\@gls@write@entryunitcounts@do}[1]{%
2853   \immediate\write\@auxout
2854   {\string\@gls@entry@unitcount
2855     {\@glsentry}%
2856     {\@glsxtr@currunitcount{\@glsentry}{#1}%
2857     }%
2858     {#1}}%
2859 }

```

entryunitcounts

```

2860 \newcommand*{\@gls@write@entryunitcounts}{%
2861   \immediate\write\@auxout
2862   {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
2863   \count@=0\relax
2864   \forallglsentries{\@glsentry}{%
2865     \gls@hasattribute{\@glsentry}{unitcount}%
2866     {%
2867       \ifglsused{\@glsentry}%
2868       {%
2869         \forlistcsloop
2870           {\@gls@write@entryunitcounts@do}%
2871           {glo@\glsdetoklabel{\@glsentry}@unitlist}%
2872       }%
2873     }%
2874     \advance\count@ by \@ne
2875   }%
2876 }%
2877 }%
2878 \ifnum\count@=0
2879   \GlossariesExtraWarningNoLine{Entry counting has been enabled

```

```

2880 \MessageBreak with \string\glsenableentryunitcount\space but the
2881 \MessageBreak attribute ‘unitcount’ hasn’t
2882 \MessageBreak been assigned to any of the defined
2883 \MessageBreak entries}%
2884 \fi
2885 }

```

`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.

```

2886 \newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%

```

Enable entry counting:

```

2887 \glsenableentryunitcount

```

Redefine `\gls` etc:

```

2888 \renewcommand*{\gls}{\cglsl}%
2889 \renewcommand*{\Gls}{\cGls}%
2890 \renewcommand*{\glspl}{\cglspl}%
2891 \renewcommand*{\Glspl}{\cGlspl}%
2892 \renewcommand*{\GLS}{\cGLS}%
2893 \renewcommand*{\GLSpl}{\cGLSpl}%

```

Set the entrycount attribute:

```

2894 \@glstxr@setentryunitcountunsetattr{#1}{#2}{#3}%

```

In case this command is used again:

```

2895 \let\GlsXtrEnableEntryUnitCounting\@glstxr@setentryunitcountunsetattr
2896 \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
2897 \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
2898 can’t be used with \string\GlsXtrEnableEntryUnitCounting}%
2899 {Use one or other but not both commands}}%
2900 }

```

`countunsetattr`

```

2901 \newcommand*{\@glstxr@setentryunitcountunsetattr}[3]{%
2902 \@for\@glstxr@cat:=#1\do
2903 {%
2904 \ifdefempty{\@glstxr@cat}{}%
2905 {%
2906 \glssetcategoryattribute{\@glstxr@cat}{entrycount}{#2}%
2907 \glssetcategoryattribute{\@glstxr@cat}{unitcount}{#3}%
2908 }%
2909 }%
2910 }

```

1.3.6 Acronym Modifications

It’s more consistent to use the abbreviation code for acronyms, but make some adjustments to allow for continued use of the glossaries package’s custom acronym format. (For example, user may already have defined some acronym styles with `\newacronymstyle` which they

would like to continue to use.) The original glossaries acronym code can be restored with `\RestoreAcronyms`, but adjust `\SetGenericNewAcronym` so that `\newacronym` adds the category.

`GenericNewAcronym`

```

2911 \renewcommand*{\SetGenericNewAcronym}{%
2912   \let\@Gls@entryname\@Gls@acentryname
2913   \renewcommand{\newacronym}[4][{}]{%
2914     \ifdefempty{\@glsacronymlists}%
2915     {%
2916       \def\@glo@type{\acronymtype}%
2917       \setkeys{glossentry}{##1}%
2918       \DeclareAcronymList{\@glo@type}%
2919     }%
2920   }%
2921   \glskeylisttok{##1}%
2922   \glslabeltok{##2}%
2923   \glsshorttok{##3}%
2924   \glslongtok{##4}%
2925   \newacronymhook
2926   \protected@edef\@do@newglossaryentry{%
2927     \noexpand\newglossaryentry{\the\glslabeltok}%
2928     {%
2929       type=\acronymtype,%
2930       name={\expandonce{\acronymentry{##2}}},%
2931       sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
2932       text={\the\glsshorttok},%
2933       short={\the\glsshorttok},%
2934       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
2935       long={\the\glslongtok},%
2936       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
2937       category=acronym,%
2938       \GenericAcronymFields,%
2939       \the\glskeylisttok
2940     }%
2941   }%
2942   \@do@newglossaryentry
2943 }%
2944 \renewcommand*{\acrfullfmt}[3]{%
2945   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
2946 \renewcommand*{\Acrfullfmt}[3]{%
2947   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
2948 \renewcommand*{\ACRfullfmt}[3]{%
2949   \glslink[##1]{##2}{%
2950     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
2951 \renewcommand*{\acrfullplfmt}[3]{%
2952   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
2953 \renewcommand*{\Acrfullplfmt}[3]{%
2954   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%

```

```

2955 \renewcommand*{\ACRfullplfmt}[3]{%
2956   \glslink{##1}{##2}}{%
2957   \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%
2958 \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
2959 \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
2960 \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
2961 \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
2962 }

```

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:

```

2963 \let\@glxtr@org@setacronymstyle\setacronymstyle
2964 \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`.

```

2965 \newcommand*{\MakeAcronymsAbbreviations}{%
2966   \renewcommand*{\newacronym}[4][1]{%
2967     \glxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
2968   }%
2969   \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
2970   \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
2971   \renewcommand*{\setacronymstyle}[1]{%
2972     \PackageError{glossaries-extra}{\string\setacronymstyle{##1}}
2973     unavailable.
2974     Use \string\setabbreviationstyle\space instead.
2975     The original acronym interface can be restored with
2976     \string\RestoreAcronyms}{}%
2977   }%
2978   \renewcommand*{\newacronymstyle}[1]{%
2979     \GlossariesExtraWarning{New acronym style ‘##1’ won’t be
2980     available unless you restore the original acronym interface with
2981     \string\RestoreAcronyms}%
2982     \@glxtr@org@newacronymstyle{##1}%
2983   }%
2984 }

```

Switch acronyms to abbreviations:

```

2985 \MakeAcronymsAbbreviations

```

RestoreAcronyms Restore acronyms to glossaries interface.

```

2986 \newcommand*{\RestoreAcronyms}{%
2987   \SetGenericNewAcronym
2988   \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
2989   \renewcommand*{\acronymfont}[1]{##1}%
2990   \let\setacronymstyle\@glxtr@org@setacronymstyle
2991   \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.

```

2992 \renewcommand*\@gls@link@checkfirsthyper{%
2993   \ifglslused{\glslabel}%
2994   {\let\glxtrifwasfirstuse\@secondoftwo}
2995   {\let\glxtrifwasfirstuse\@firstoftwo}%
2996   \@glxtr@org@checkfirsthyper
2997 }
2998 \glssetcategoryattribute{acronym}{regular}{false}%
2999 \setacronymstyle{long-short}%
3000 }

```

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

```

3001 \renewcommand*\glsacspace[1]{%
3002   \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
3003   \ifdim\dimen@<\glsacspacemax~\else\space\fi
3004 }

```

`\glsacspacemax` Value used in the above.

```

3005 \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`

```

3006 \newcommand*\@glxtr@reg@glosslist{}

```

Save the original definition of `\makeglossaries`:

```

3007 \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`

```

3008 \renewcommand*\makeglossaries[1][]{%
3009   \ifblank{#1}%
3010   {\@glxtr@org@makeglossaries}%
3011   {%
3012     \edef\@glxtr@reg@glosslist{#1}%
3013     \ifundef{\glswrite}{\newwrite\glswrite}{}%
3014     \protected@write\@auxout{}\string\providecommand
3015       \string\@glorder[1]{}
3016     \protected@write\@auxout{}\string\providecommand

```



```

3017 \string\@istfilename[1]{}%
3018 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
3019 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}
3020 \protected@write\@auxout{}{\string\glsxtr@makeglossaries{#1}}
3021 \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%

```

Iterate through each supplied glossary type and activate it.

```

3022 \@for\@glo@type:=#1\do{%
3023 \ifdefempty{\@glo@type}{}{\@makeglossary{\@glo@type}}%
3024 }%

```

New glossaries must be created before \makeglossaries:

```

3025 \renewcommand*\newglossary[4][]{%
3026 \PackageError{glossaries}{New glossaries
3027 must be created before \string\makeglossaries}{You need
3028 to move \string\makeglossaries\space after all your
3029 \string\newglossary\space commands}}%

```

Any subsequent instances of this command should have no effect

```

3030 \let\@makeglossary\relax
3031 \let\makeglossary\relax
3032 \renewcommand\makeglossaries[1][]{}%

```

Disable all commands that have no effect after \makeglossaries

```

3033 \@disable@onlypremakeg

```

Allow see key:

```

3034 \let\gls@checkseeallowed\relax

```

Adjust \@do@seeglossary

```

3035 \renewcommand*\@do@seeglossary[2]{%
3036 \edef\@gls@label{\glsdetoklabel{##1}}%
3037 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3038 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3039 {\@glsxtr@org@doseeglossary{##1}{##2}}%
3040 {%
3041 \protected@write\@auxout{}{%
3042 \string\@gls@reference
3043 {\@gls@type}{\@gls@label}{\string\glsseeformat##2{}}%
3044 }%
3045 }%
3046 }%

```

Adjust \@do@wrglossary

```

3047 \let\glsxtr@do@wrglossary\@do@wrglossary
3048 \def\@do@wrglossary{%
3049 \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
3050 \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3051 {\@glsxtr@do@wrglossary}%
3052 {\@gls@noidxglossary}%
3053 }%

```

Suppress warning about no \makeglossaries

```
3054 \let\warn@nomakeglossaries\relax
3055 \def\warn@noprintglossary{%
3056   \GlossariesWarningNoLine{No \string\printglossary\space
3057     or \string\printglossaries\space
3058     found.^^J(Remove \string\makeglossaries\space if you don't want
3059     any glossaries.)^^JThis document will not have a glossary}%
3060 }%
```

Only warn for glossaries not listed.

```
3061 \renewcommand{\@gls@noref@warn}[1]{%
3062   \edef\@gls@type{##1}%
3063   \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
3064   {%
3065     \GlossariesExtraWarning{Can't use
3066       \string\printnoidxglossary[type={\@gls@type}]
3067       when '@gls@type' is listed in the optional argument of
3068       \string\makeglossaries}%
3069   }%
3070   {%
3071     \GlossariesWarning{Empty glossary for
3072       \string\printnoidxglossary[type={##1}].
3073       Rerun may be required (or you may have forgotten to use
3074       commands like \string\gls)}%
3075   }%
3076 }%
```

Adjust display number list to check for type:

```
3077 \renewcommand*\@glsdisplaynumberlist[1]{%
3078   \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3079   {\@glsxtr@idx@displaynumberlist{##1}}%
3080   {\@glsxtr@noidx@displaynumberlist{##1}}%
3081 }%
```

Adjust entry list:

```
3082 \renewcommand*\@glsentrynumberlist[1]{%
3083   \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3084   {\@glsxtr@idx@entrynumberlist{##1}}%
3085   {\@glsxtr@noidx@entrynumberlist{##1}}%
3086 }%
```

Adjust number list loop

```
3087 \renewcommand*\@glsnumberlistloop[2]{%
3088   \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
3089   {%
3090     \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
3091       not available for glossary '##1'}{%
3092     }%
3093     {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
3094   }%
```

Only sanitize sort for normal indexing glossaries.

```
3095 \renewcommand*{\glsprestandardsort}[3]{%
3096   \expandafter\DTLifinlist\expandafter{##2}{\@glxtr@reg@glosslist}%
3097   {%
3098     \glsdosanitizesort
3099   }%
3100   {%
3101     \ifglssanitizesort
3102       \@gls@noidx@sanitizesort
3103     \else
3104       \@gls@noidx@nosanitizesort
3105     \fi
3106   }%
3107 }%
```

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

```
3108 \renewcommand*{\new@glossaryentry}[2]{%
3109   \PackageError{glossaries-extra}{Glossary entries must be defined
3110     in the preamble\MessageBreak when you use the optional argument
3111     of \string\makeglossaries}{Either move your definitions to the
3112     preamble or don't use the optional argument of
3113     \string\makeglossaries}%
3114 }%
```

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).

```
3115 \let\@glo@assign@sortkey\@glxtr@mixed@assign@sortkey
3116 \renewcommand*{\@printgloss@setsort}{%
```

Need to extract just the type value.

```
3117   \expandafter\@glxtr@gettype\expandafter,\@glxtr@printglossopts,%
3118   type=\glsdefaulttype,\@end@glxtr@gettype
3119   \def\@glo@sorttype{\@glo@default@sorttype}%
3120 }%
```

Check automake setting:

```
3121 \ifglssautomake
3122   \renewcommand*{\@gls@doautomake}{%
3123     \@for\@gls@type:=\@glxtr@reg@glosslist\do{%
3124       \ifdefempty{\@gls@type}{\@gls@automake{\@gls@type}}%
3125     }%
3126   }%
3127 \fi
3128 }%
3129 }
```

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

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

```
3130 \newcommand{\@glxtr@orgprintglossary}[2]{%
3131   \def\@glo@type{\glstexttype}%
```

Add check here.

```
3132   \def\glossarytitle{%
3133     \ifcsdef{\@glo@type\@glo@type @title}%
3134     {\csuse{\@glo@type\@glo@type @title}}%
3135     {\glossaryname}}%
3136   \def\glossarytoctitle{\glossarytitle}%
3137   \let\org@glossarytitle\glossarytitle
3138   \def\@glossarystyle{%
3139     \ifx\@glossary@default@style\relax
3140       \GlossariesWarning{No default glossary style provided \MessageBreak
3141         for the glossary '\@glo@type'. \MessageBreak
3142         Using deprecated fallback. \MessageBreak
3143         To fix this set the style with \MessageBreak
3144         \string\setglossarystyle\space or use the \MessageBreak
3145         style key=value option}%
3146     \fi
3147   }%
3148   \def\gls@dotoc@title{\gls@settoc@title{\@glo@type}}%
3149   \let\org@glossaryentrynumbers\glossaryentrynumbers
3150   \bgroup
3151     \@printgloss@setsort
3152     \setkeys{printgloss}{#1}%
3153     \ifx\glossarytitle\org@glossarytitle
3154     \else
3155       \cslet{\@glo@type\@glo@type @title}{\glossarytitle}%
3156     \fi
3157     \let\currentglossary\@glo@type
3158     \let\org@glossaryentrynumbers\glossaryentrynumbers
3159     \let\glsnonextpages\@glsnonextpages
3160     \let\glsnextpages\@glsnextpages
3161     \let\nopostdesc\@nopostdesc
3162     \gls@dotoc@title
3163     \@glossarystyle
3164     \let\gls@org@glossaryentryfield\glossentry
3165     \let\gls@org@glossarysubentryfield\subglossentry
3166     \renewcommand{\glossentry}[1]{%
3167       \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
3168       \gls@org@glossaryentryfield{##1}%
3169     }%
3170     \renewcommand{\subglossentry}[2]{%
3171       \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
3172       \gls@org@glossarysubentryfield{##1}{##2}%
```

```

3173 }%
3174 \@gls@preglossaryhook
3175 #2%
3176 \egroup
3177 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
3178 \global\let\warn@noprintglossary\relax
3179 }

```

`\@printglossary` Redefine.

```

3180 \renewcommand{\@printglossary}[2]{%
3181   \def\@glsxtr@printglossopts{#1}%
3182   \@glsxtr@orgprintglossary{#1}{#2}%
3183 }

```

Add a key that switches off the entry targets:

```

3184 \define@choicekey{printgloss}{target}[\val\nr]{true,false}[true]{%
3185   \ifcase\nr
3186     \let\@glstarget\glsdohypertarget
3187   \else
3188     \let\@glstarget\@secondoftwo
3189   \fi
3190 }

```

`@makeglossaries` For the benefit of makeglossaries

```

3191 \newcommand*{\@glsxtr@makeglossaries}[1]{%

```

`@glsxtr@gettype` Get just the type.

```

3192 \def\@glsxtr@gettype#1,type=#2,#3\@end@glsxtr@gettype{%
3193   \def\@glo@type{#2}%
3194 }

```

`@assign@sortkey` Assign the sort key.

```

3195 \newcommand\@glsxtr@mixed@assign@sortkey[1]{%
3196   \edef\@glo@type{\@glo@type}%
3197   \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glsxtr@reg@glosslist}%
3198   {%
3199     \@glo@no@assign@sortkey{#1}%
3200   }%
3201   {%
3202     \@glo@assign@sortkey{#1}%
3203   }%
3204 }%

```

Display number list for the regular version:

`splaynumberlist`

```

3205 \let\@glsxtr@idx@displaynumberlist\glsdisplaynumberlist

```

Display number list for the “noidx” version:

splaynumberlist

```

3206 \newcommand*{\@glsxtr@noidx@displaynumberlist}[1]{%
3207   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
3208   \ifdef\@gls@loclist
3209   {%
3210     \def\@gls@noidxloclist@sep{%
3211       \def\@gls@noidxloclist@sep{%
3212         \def\@gls@noidxloclist@sep{%
3213           \glsnumlistsep
3214         }%
3215       \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
3216     }%
3217   }%
3218   \def\@gls@noidxloclist@finalsep{}%
3219   \def\@gls@noidxloclist@prev{}%
3220   \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
3221   \@gls@noidxloclist@finalsep
3222   \@gls@noidxloclist@prev
3223 }%
3224 {%
3225   ??\glsdoifexists{#1}%
3226   {%
3227     \GlossariesWarning{Missing location list for ‘#1’. Either
3228       a rerun is required or you haven’t referenced the entry.}%
3229   }%
3230 }%
3231 }%
3232

```

And for the number list loop:

@numberlistloop

```

3233 \newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
3234   \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
3235   \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
3236   \let\@gls@org@glsseeformat\glsseeformat
3237   \let\glsnoidxdisplayloc#2\relax
3238   \let\glsseeformat#3\relax
3239   \ifdef\@gls@loclist
3240   {%
3241     \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
3242   }%
3243   {%
3244     ??\glsdoifexists{#1}%
3245     {%
3246       \GlossariesWarning{Missing location list for ‘##1’. Either
3247         a rerun is required or you haven’t referenced the entry.}%
3248     }%
3249   }%
3250   \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc

```

```

3251 \let\glsseeformat\@gls@org@glsseeformat
3252 }%

```

Same for entry number list.

entrynumberlist

```

3253 \newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
3254 \letcs{\@gls@loclist}{glo\@glsdetoklabel{#1}@loclist}%
3255 \ifdef\@gls@loclist
3256 {%
3257 \glsnoidxloclist{\@gls@loclist}%
3258 }%
3259 {%
3260 ??\glsdoifexists{#1}%
3261 {%
3262 \GlossariesWarning{Missing location list for ‘#1’. Either
3263 a rerun is required or you haven’t referenced the entry.}%
3264 }%
3265 }%
3266 }%

```

entrynumberlist

```

3267 \newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\@glsentrynumberlist{#1}}

```

x@getgrouptitle Patch.

```

3268 \renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
3269 \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
3270 \@onelevel@sanitize\@glsxtr@titlelabel
3271 \ifcsundef{\@glsxtr@titlelabel}%
3272 {%
3273 \DTLifint{#1}%
3274 {%
3275 \ifnum#1<256\relax
3276 \edef#2{\char#1\relax}%
3277 \else
3278 \edef#2{#1}%
3279 \fi
3280 }%
3281 {%
3282 \ifcsundef{#1groupname}%
3283 {\def#2{#1}}%
3284 {\letcs#2{#1groupname}}%
3285 }%
3286 }%
3287 {%
3288 \letcs#2{\@glsxtr@titlelabel}%
3289 }%
3290 }

```

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

```

3291 \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.

```

3292 \newrobustcmd{\glsxtrgetgrouptitle}[2]{%
3293   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
3294   \@onelevel@sanitize\@glsxtr@titlelabel
3295   \ifcsdef{\@glsxtr@titlelabel}
3296     {\letcs{#2}{\@glsxtr@titlelabel}}%
3297   {\glsxtr@org@getgrouptitle{#1}{#2}}%
3298 }
3299 \let\@gls@getgrouptitle\glsxtrgetgrouptitle

```

`trsetgrouptitle` Sets the title for the given group label.

```

3300 \newcommand{\glsxtrsetgrouptitle}[2]{%
3301   \protected@edef\@glsxtr@titlelabel{\glsxtr@grouptitle@#1}%
3302   \@onelevel@sanitize\@glsxtr@titlelabel
3303   \csxdef{\@glsxtr@titlelabel}{#2}%
3304 }

```

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

```

3305 \renewcommand*{\glsnavigation}{%
3306   \def\@gls@between{}%
3307   \ifcsundef{\@gls@hypergroup@list@{\@glo@type}}%
3308     {%
3309       \def\@gls@list{}%
3310     }%
3311     {%
3312       \expandafter\let\expandafter\@gls@list
3313         \csname @gls@hypergroup@list@{\@glo@type}\endcsname
3314     }%
3315     \@for\@gls@tmp:=\@gls@list\do{%
3316       \@gls@between
3317       \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
3318       \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
3319       \let\@gls@between\glshypernavsep
3320     }%
3321 }

```

`@noidx@glossary`

```

3322 \renewcommand*{\@print@noidx@glossary}{%
3323   \ifcsdef{\@glsref@{\@glo@type}}%
3324     {%
3325       \ifcsdef{\@glo@sortmacro@{\@glo@sorttype}}%
3326         {%
3327           \csuse{\@glo@sortmacro@{\@glo@sorttype}}{\@glo@type}%
3328         }%

```



```

3329  {%
3330      \PackageError{glossaries}{Unknown sort handler ‘\@glo@sorttype’}{}%
3331  }%
3332  \glossarysection[\glossarytoctitle]{\glossarytitle}%
3333  \glossarypreamble

```

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

```

3334  \def\@gls@currentlettergroup{%
3335  \begin{theglossary}%
3336  \glossaryheader
3337  \glsresetentrylist
3338  \forlistcsloop{\@gls@noidx@do}{\@glsref@{\@glo@type}%
3339  \end{theglossary}%
3340  \glossarypostamble
3341  }%
3342  {%

```

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

```

3343  \glsxtrifemptyglossary{\@glo@type}%
3344  {}%
3345  {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
3346  \@gls@noref@warn{\@glo@type}%
3347  }%
3348 }

```

noidxdisplayloc Patch to check for range formations.

```

3349 \renewcommand*{\glsnoidxdisplayloc}[4]{%
3350 \setentrycounter[#1]{#2}%
3351 \@glsxtr@display@loc#3\empty\end@glxtr@display@loc{#4}%
3352 }

```

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

```

3353 \def\@glsxtr@display@loc#1#2\end@glxtr@display@loc#3{%
3354 \ifx#1(\relax
3355 \glsxtrdisplaystartloc{#2}{#3}%
3356 \else
3357 \ifx#1)\relax
3358 \glsxtrdisplayendloc{#2}{#3}%
3359 \else
3360 \glsxtrdisplaysingleloc{#1#2}{#3}%
3361 \fi
3362 \fi
3363 }

```

isplaysingleloc Single location.

```

3364 \newcommand*{\glsxtrdisplaysingleloc}[2]{%
3365 \csuse{#1}{#2}%
3366 }

```

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.

```
3367 \newcommand*{\glxtrdisplaystartloc}[2]{%
3368   \edef\glxtrlocrangefmt{#1}%
3369   \ifx\glxtrlocrangefmt\empty
3370     \def\glxtrlocrangefmt{glxnumberformat}%
3371   \fi
3372   \expandafter\glxtrdisplaysingleloc
3373   \expandafter{\glxtrlocrangefmt}{#2}%
3374 }
```

`trdisplayendloc` End of a location range.

```
3375 \newcommand*{\glxtrdisplayendloc}[2]{%
3376   \edef\@glxtr@tmp{#1}%
3377   \ifdefempty{\@glxtr@tmp}{\def\@glxtr@tmp{glxnumberformat}}{}}%
3378   \ifx\glxtrlocrangefmt\@glxtr@tmp
3379   \else
3380     \GlossariesExtraWarning{Mismatched end location range
3381       (start=\glxtrlocrangefmt, end=\@glxtr@tmp)}%
3382   \fi
3383   \expandafter\glxtrdisplayendlochook\expandafter{\@glxtr@tmp}{#2}%
3384   \expandafter\glxtrdisplaysingleloc
3385   \expandafter{\glxtrlocrangefmt}{#2}%
3386   \def\glxtrlocrangefmt{}%
3387 }
```

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

```
3388 \newcommand*{\glxtrdisplayendlochook}[2]{}
```

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

```
3389 \newcommand*{\glxtrlocrangefmt}{}%
```

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

```
3390 \def\@gls@removespaces#1 #2\@nil{%
3391   \toks@=\expandafter{\the\toks@#1}%
3392   \ifx\@#2\%
3393     \edef\x{\the\toks@}%
3394     \ifx\x\empty
3395     \else
3396       \glxtrlocationhyperlink{\glxentrycounter}{\@glo@counterprefix}{\the\toks@}%
3397     \fi
3398   \else
3399     \@gls@ReturnAfterFi{%
3400       \@gls@removespaces#2\@nil
3401     }%
3402   \fi
3403 }
```

cationhyperlink

```
3404 \newcommand*{\glxtrlocationhyperlink}[3]{%
3405   \ifdefvoid\glxtrsupplocationurl
3406   {%
3407     \hyperlink{#1#2#3}{#3}%
3408   }%
3409   {%
3410     \hyperref{\glxtrsupplocationurl}{#1#2#3}{#3}%
3411   }%
3412 }
```

supphypernumber

```
3413 \newcommand*{\glxtrsupphypernumber}[1]{%
3414   {%
3415     \glshasattribute{\glscurrententrylabel}{externallocation}%
3416     {%
3417       \def\glxtrsupplocationurl{%
3418         \glsggetattribute{\glscurrententrylabel}{externallocation}}%
3419     }%
3420     {%
3421       \def\glxtrsupplocationurl{}%
3422     }%
3423     \glshypernumber{#1}%
3424   }%
3425 }
```

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

@print@glossary

```
3426 \renewcommand{\@print@glossary}{%
3427   \makeatletter
3428   \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
3429   \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
3430   {%
3431     {\glxtrNoGlossaryWarning{\@glo@type}}%
3432     \ifglxindy
3433       \ifcsundef{@xdy@\@glo@type @language}%
3434       {%
3435         \edef\@do@auxoutstuff{%
3436           \noexpand\AtEndDocument{%
3437             \noexpand\immediate\noexpand\write\@auxout{%
3438               \string\providecommand\string\@xdylanguage[2]{}}%
3439             \noexpand\immediate\noexpand\write\@auxout{%
3440               \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
3441           }%
3442         }%
3443       }%
3444     {%
```

```

3445 \edef\@do@auxoutstuff{%
3446 \noexpand\AtEndDocument{%
3447 \noexpand\immediate\noexpand\write\@auxout{%
3448 \string\providecommand\string\@xdylanguage[2]{}}%
3449 \noexpand\immediate\noexpand\write\@auxout{%
3450 \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
3451 @language\endcsname}}%
3452 }%
3453 }%
3454 }%
3455 \@do@auxoutstuff
3456 \edef\@do@auxoutstuff{%
3457 \noexpand\AtEndDocument{%
3458 \noexpand\immediate\noexpand\write\@auxout{%
3459 \string\providecommand\string\@gls@codepage[2]{}}%
3460 \noexpand\immediate\noexpand\write\@auxout{%
3461 \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
3462 }%
3463 }%
3464 \@do@auxoutstuff
3465 \fi
3466 \renewcommand*{\@warn@nomakeglossaries}{%
3467 \GlossariesWarningNoLine{\string\makeglossaries\space
3468 hasn't been used,~^Jthe glossaries will not be updated}%
3469 }%
3470 }

```

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

`\GlsWarningHead` Header message.

```

3471 \newcommand{\GlsXtrNoGlsWarningHead}[2]{%
3472 This document is incomplete. The external file associated with
3473 the glossary '#1' (which should be called \texttt{#2})
3474 hasn't been created.%
3475 }

```

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

```

3476 \newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
3477 This has probably happened because there are no entries defined
3478 in this glossary.%
3479 }

```

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

```

3480 \newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
3481 If you don't want this glossary,
3482 add \texttt{nomain} to your package option list when you load
3483 \texttt{glossaries-extra.sty}. For example:%
3484 }

```

ingEmptyNotMain A glossary that isn't the default "main" glossary is empty.

```
3485 \newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
3486 Did you forget to use \texttt{type=#1} when you defined your
3487 entries? If you tried to load entries into this glossary with
3488 \texttt{\string\loadglsentries} did you remember to use
3489 \texttt{[#1]} as the optional argument? If you did, check that
3490 the definitions in the file you loaded all had the type set
3491 to \texttt{\string\glsdefaulttype}.\%
3492 }
```

arningCheckFile Advisory message to check the file contents.

```
3493 \newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
3494 Check the contents of the file \texttt{#1}. If
3495 it's empty, that means you haven't indexed any of your entries in this
3496 glossary (using commands like \texttt{\string\gls} or
3497 \texttt{\string\glsadd}) so this list can't be generated.
3498 If the file isn't empty, the document build process hasn't been
3499 completed.\%
3500 }
```

WarningAutoMake Message when automake option has been used.

```
3501 \newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
3502 You may need to rerun \LaTeX. If you already have, it may be that
3503 \TeX's shell escape doesn't allow you to run
3504 \ifglxindy xindy\else makeindex\fi. Check the
3505 transcript file \texttt{\jobname.log}. If the shell escape is
3506 disabled, try one of the following:
3507
3508 \begin{itemize}
3509 \item Run the external (Lua) application:
3510
3511 \texttt{makeglossaries-lite.lua \string"\jobname\string"}
3512
3513 \item Run the external (Perl) application:
3514
3515 \texttt{makeglossaries \string"\jobname\string"}
3516 \end{itemize}
3517
3518 Then rerun \LaTeX\ on this document.
3519 \GlossariesExtraWarning{Rerun required to build the
3520 glossary '#1' or check TeX's shell escape allows
3521 you to run \ifglxindy xindy\else makeindex\fi}%
3522 }
```

WarningMisMatch Mismatching \makenoidxglossaries.

```
3523 \newcommand{\GlsXtrNoGlsWarningMisMatch}{%
3524 You need to either replace \texttt{\string\makenoidxglossaries}
3525 with \texttt{\string\makeglossaries} or replace
```

```

3526 \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
3527 \texttt{\string\printnoidxglossary}
3528 (or \texttt{\string\printnoidxglossaries}) and then rebuild
3529 this document.%
3530 }

```

arningBuildInfo Build advice.

```

3531 \newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
3532   Try one of the following:
3533   \begin{itemize}
3534     \item Add \texttt{automake} to your package option list when you load
3535           \texttt{glossaries-extra.sty}. For example:
3536
3537           \texttt{\string\usepackage[automake]%
3538                 \glsopenbrace glossaries-extra\glsclosebrace}
3539
3540     \item Run the external (Lua) application:
3541
3542           \texttt{makeglossaries-lite.lua \string"\jobname\string"}
3543
3544     \item Run the external (Perl) application:
3545
3546           \texttt{makeglossaries \string"\jobname\string"}
3547   \end{itemize}
3548
3549   Then rerun \LaTeX\ on this document.%
3550 }

```

oGlsWarningTail Final paragraph.

```

3551 \newcommand{\GlsXtrNoGlsWarningTail}{%
3552   This message will be removed once the problem has been fixed.%
3553 }

```

GlsWarningNoOut No out file created. Build advice.

```

3554 \newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
3555   The file \texttt{#1} doesn't exist. This most likely means you haven't used
3556   \texttt{\string\makeglossaries} or you have used
3557   \texttt{\string\nofiles}. If this is just a draft version of the
3558   document, you can suppress this message using the
3559   \texttt{nomissingglstext} package option.%
3560 }

```

glossarywarning

```

3561 \newcommand*{@@glxtr@defaultnoglossarywarning}[1]{%
3562   \glossarysection[\glossarytoctitle]{\glossarytitle}
3563   \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
3564   \par
3565   \glxtrifemptyglossary{#1}%
3566   {%

```

```

3567 \GlsXtrNoGlsWarningEmptyStart\space
3568 \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
3569 \medskip
3570 \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
3571 \glsopenbrace glossaries-extra\glsclosebrace}
3572 \medskip
3573 }%
3574 {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
3575 }%
3576 {%
3577 \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
3578 {%
3579 \GlsXtrNoGlsWarningCheckFile
3580 {\jobname.\csname @glotype@\@glo@type @out\endcsname}
3581
3582 \ifglsautomake
3583
3584 \GlsXtrNoGlsWarningAutoMake{#1}
3585
3586 \else
3587
3588 \ifthenelse{\equal{#1}{main}}%
3589 {%
3590 \GlsXtrNoGlsWarningEmptyMain\par
3591 \medskip
3592 \noindent\texttt{\string\usepackage[nomain]%
3593 \glsopenbrace glossaries-extra\glsclosebrace}
3594 \medskip
3595 }%
3596 {}%
3597
3598 \ifdefequal\makeglossaries\@no@makeglossaries
3599 {%
3600 \GlsXtrNoGlsWarningMisMatch
3601 }%
3602 {%
3603 \GlsXtrNoGlsWarningBuildInfo
3604 }%
3605 \fi
3606 }%
3607 {%
3608 \GlsXtrNoGlsWarningNoOut
3609 {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
3610 }%
3611 }%
3612 \par
3613 \GlsXtrNoGlsWarningTail
3614 }

```

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

xttrresourcefile 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.

```

3615 \newcommand*{\glxtrresourcefile}[2] [] {%
3616   \protected@write\@auxout{}\string\glxtr@resource{#1}{#2}}%
3617   \glxtr@writefields
3618   \let\@glxtr@org@see@noindex\@glxtr@see@noindex
3619   \let\@glxtr@see@noindex\relax
3620   \IfFileExists{#2.glstex}%
3621   {%

```

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

```

3622     \edef\@bibglxtr@restoreat{\noexpand\catcode\noexpand'\noexpand\@=\number\catcode'\@}%
3623     \makeatletter
3624     \@input{#2.glstex}%
3625     \@bibglxtr@restoreat
3626   }%
3627   {%
3628     \GlossariesExtraWarning{No file '#2.glstex'}%
3629   }%
3630   \let\@glxtr@see@noindex\@glxtr@org@see@noindex
3631 }
3632 \@onlypreamble\glxtrresourcefile

```

trresourcecount

```

3633 \newcount\glxtrresourcecount

```

trLoadResources Short cut that uses \glxtrresourcefile with \jobname as the mandatory argument.

```

3634 \newcommand*{\GlsXtrLoadResources}[1] [] {%
3635   \ifnum\glxtrresourcecount=0\relax
3636   \glxtrresourcefile[#1]{\jobname}%
3637   \else
3638   \glxtrresourcefile[#1]{\jobname-\the\glxtrresourcecount}%
3639   \fi
3640   \advance\glxtrresourcecount by 1\relax
3641 }

```

glxtr@resource

```

3642 \newcommand*{\glxtr@resource}[2] {}

```

\glxtr@fields

```

3643 \newcommand*{\glxtr@fields}[1] {}

```

xttr@texencoding

```

3644 \newcommand*{\glxtr@texencoding}[1] {}

```

\glxtr@langtag

```

3645 \newcommand*{\glxtr@langtag}[1] {}

```


@pluralsuffixes

```
3646 \newcommand*{\glxtr@pluralsuffixes}[4]{}%
```

tr@shortcutsval

```
3647 \newcommand*{\glxtr@shortcutsval}[1]{}%
```

sxtr@linkprefix

```
3648 \newcommand*{\glxtr@linkprefix}[1]{}%
```

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```
3649 \newcommand*{\glxtr@writefields}{%
```

```
3650   \protected@write\@auxout{}\string\glxtr@fields{\@gls@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 \glxtrresourcefile.

```
3651   \ifdef\CurrentTrackedLanguageTag
```

```
3652   {%
```

```
3653     \protected@write\@auxout{}\string\glxtr@langtag{\CurrentTrackedLanguageTag}}%
```

```
3654     \string\glxtr@langtag{\CurrentTrackedLanguageTag}}%
```

```
3655   }%
```

```
3656   }%
```

```
3657   \protected@write\@auxout{}\string\glxtr@pluralsuffixes
```

```
3658     {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}}%
```

```
3659     {\glxtrabbrvpluralsuffix}}%
```

```
3660   \ifdef\inputencodingname
```

```
3661   {%
```

```
3662     \protected@write\@auxout{}\string\glxtr@texencoding{\inputencodingname}}%
```

```
3663   }%
```

```
3664   }%
```

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.)

```
3665     \@ifpackageloaded{fontspec}%
```

```
3666     {\protected@write\@auxout{}\string\glxtr@texencoding{utf8}}}%
```

```
3667     {}%
```

```
3668   }%
```

```
3669   \protected@write\@auxout{}\string\glxtr@shortcutsval{\@glxtr@shortcutsval}}%
```

Prefix deferred until the beginning of the document in case it's redefined later in the preamble. This is picked up by bib2gls when the external option is used.

```
3670   \AtBeginDocument
```

```
3671     {\protected@write\@auxout{}\string\glxtr@linkprefix{\glolinkprefix}}%
```

```
3672   \let\glxtr@writefields\relax
```

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

```
3673   \ifglautomake
```

```
3674     \IfFileExists{\jobname.aux}%
```

```
3675     {\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`.

```
3676 \ifx\@gls@doautomake\@gls@doautomake@err
3677 \let\@gls@doautomake\relax
3678 \fi
3679 \fi
3680 }
```

`do@automake@err`

```
3681 \newcommand*{\@gls@doautomake@err}{%
3682 \PackageError{glossaries}{You must use
3683 \string\makeglossaries\space with automake=true}
3684 {%
3685 Either remove the automake=true setting or
3686 add \string\makeglossaries\space to your document preamble.%
3687 }%
3688 }
```

Allow locations specific to a particular counter to be recorded.

`\glxtr@record`

```
3689 \newcommand*{\glxtr@record}[5]{}
```

`r@counterrecord` Aux file command.

```
3690 \newcommand*{\glxtr@counterrecord}[3]{%
3691 \glxtrfieldlistgadd{#1}{record.#2}{#3}%
3692 }
```

`unterrecordhook` Hook used by `\@glxtr@dorecord`.

```
3693 \newcommand*{\@glxtr@counterrecordhook}{}%
```

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

```
3694 \newcommand*{\GlsXtrRecordCounter}[1]{%
3695 \@glxtr@recordcounter{#1}%
3696 }
3697 \@onlypreamble\GlsXtrRecordCounter
```

`docounterrecord`

```
3698 \newcommand*{\@glxtr@docounterrecord}[1]{%
3699 \protected@write\@auxout{}{\string\glxtr@counterrecord
3700 {\@gls@label}{#1}{\csuse{the#1}}}%
3701 }
```

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

```
3702 \newcommand*{\printunsrtglossary}{%
3703 \@ifstar\s@printunsrtglossary\@printunsrtglossary
3704 }
```

ntunsrtglossary Unstarred version.

```
3705 \newcommand*{\@printunsrtglossary}[1][{}]{%
3706   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
3707 }
```

ntunsrtglossary Starred version.

```
3708 \newcommand*{\s@printunsrtglossary}[2][{}]{%
3709   \begingroup
3710     #2%
3711   \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
3712   \endgroup
3713 }
```

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

```
3714 \newcommand*{\printunsrtglossaries}{%
3715   \forallglossaries{\@glo@type}{\printunsrtglossary[type=\@glo@type]}%
3716 }
```

@unsrt@glossary

```
3717 \newcommand*{\@print@unsrt@glossary}{%
3718   \glossarysection[\glossarytoctitle]{\glossarytitle}%
3719   \glossarypreamble
3720   check for empty list
3721   \glsxtrifemptyglossary{\@glo@type}%
3722   {%
3723     \GlossariesExtraWarning{No entries defined in glossary ‘\@glo@type’}%
3724   }%
3725   \key@ifundefined{glossentry}{group}%
3726   {\let\@gls@getgrouptitle\@glsxtr@noidx@getgrouptitle}%
3727   {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
3728   \begin{theglossary}%
3729     \glossaryheader
3730     \glsresetentrylist
3731     \def\@gls@currentlettergroup{}%
3732     \expandafter\@for\expandafter\glscurrententrylabel\expandafter
3733       :\expandafter=\csname glo@list@\@glo@type\endcsname\do{%
3734       \ifdefempty{\glscurrententrylabel}
3735       }%
3736       {\printunsrtglossaryhandler\glscurrententrylabel}%
3737     }%
3738     \end{theglossary}%
3739   }%
3740   \glossarypostamble
3741 }
```

glossaryhandler

```

3742 \newcommand{\printunsrtglossaryhandler}[1]{%
3743   \glstrunsrtdo{#1}%
3744 }

```

srtglossaryunit

```

3745 \newcommand{\print@op@unsrtglossaryunit}[2][{}]{%
3746   \s@printunsrtglossary[type=\glsdefaulttype,#1]{%
3747     \printunsrtglossaryunitsetup{#2}%
3748   }%
3749 }

```

ossaryunitsetup

```

3750 \newcommand*{\printunsrtglossaryunitsetup}[1]{%
3751   \renewcommand{\printunsrtglossaryhandler}[1]{%
3752     \glstrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
3753     {\glstrunsrtdo{##1}}%
3754   }%
3755 }%
3756 \ifcsundef{theH#1}%
3757 {%
3758   \renewcommand*{\glolinkprefix}{record.#1.\csuse{the#1}.}%
3759 }%
3760 {%
3761   \renewcommand*{\glolinkprefix}{record.#1.\csuse{theH#1}.}%
3762 }%
3763 \renewcommand*{\glossarysection}[2][{}]{%
3764   \appto\glossarypostamble{\glspare\medskip\glspare}%
3765 }

```

srtglossaryunit

```

3766 \newcommand{\print@noop@unsrtglossaryunit}[2][{}]{%
3767   \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
3768     requires the record=only or record=alsoindex package option}{}%
3769 }

```

t@getgrouptitle

```

3770 \newrobustcmd*{\@glstr@unsrt@getgrouptitle}[2]{%
3771   \protected@edef\@glstr@titlelabel{glstr@grouptitle@#1}%
3772   \@onelevel@sanitize\@glstr@titlelabel
3773   \ifcsdef{\@glstr@titlelabel}
3774     {\letcs{#2}{\@glstr@titlelabel}}%
3775     {\def#2{#1}}%
3776 }

```

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

```

3777 \newcommand{\glstrunsrtdo}{\@glstr@noidx@do}

```

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

```

3778 \newcommand{\@glsxtr@noidx@do}[1]{%
3779   \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
3780   \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@location}%
3781   \ifglshasparent{#1}%
3782   {%
3783     \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
3784     \ifdefvoid{\@gls@location}%
3785     {%
3786       \ifdefvoid{\@gls@loclist}%
3787       {%
3788         \subglossentry{\gls@level}{#1}{}%
3789       }%
3790     }%
3791     \subglossentry{\gls@level}{#1}%
3792     {%
3793       \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
3794     }%
3795   }%
3796 }%
3797 {%
3798   \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
3799 }%
3800 }%
3801 {%
3802   \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
3803   \key@ifundefined{glossentry}{group}%
3804   {%
3805     \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
3806   }%
3807   {%
3808     \protected@xdef\@glo@thislettergrp{%
3809       \csname glo@\glsdetoklabel{#1}@group\endcsname}%
3810   }%
3811   \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
3812   {}%
3813   {%
3814     \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
3815     \expandafter\gls@groupheading\expandafter
3816       {\csname glo@\glsdetoklabel{#1}@group\endcsname}%
3817   }%
3818   \let\@gls@currentlettergroup\@glo@thislettergrp
3819   \ifdefvoid{\@gls@location}%
3820   {%
3821     \ifdefvoid{\@gls@loclist}
3822     {%
3823       \glossentry{#1}{}%
3824     }%
3825     {%
3826       \glossentry{#1}%

```

```

3827      {%
3828      \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}}%
3829      }%
3830    }%
3831  }%
3832  {%
3833    \glossentry{#1}%
3834    {%
3835      \glossaryentrynumbers{\@gls@location}%
3836    }%
3837  }%
3838 }%
3839 }

```

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.

```

3840 \@ifpackageloaded{glossaries-accsupp}
3841 {

```

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

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

```

3842 \newcommand*{\glsaccessname}[1]{%
3843 \glsnameaccessdisplay
3844 {%
3845 \glsentryname{#1}%
3846 }%
3847 {#1}%
3848 }

```

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

```

3849 \newcommand*{\Glsaccessname}[1]{%
3850 \glsnameaccessdisplay
3851 {%
3852 \Glsentryname{#1}%
3853 }%
3854 {#1}%
3855 }

```

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

```

3856 \newcommand*{\GLSaccessname}[1]{%
3857 \glsnameaccessdisplay
3858 {%

```

```

3859      \mfirstucMakeUppercase{\glsentryname{#1}}%
3860    }%
3861    {#1}%
3862  }

```

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

```

3863  \newcommand*{\glsaccesstext}[1]{%
3864    \glstextaccessdisplay
3865    {%
3866      \glsentrytext{#1}%
3867    }%
3868    {#1}%
3869  }

```

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

```

3870  \newcommand*{\Glsaccesstext}[1]{%
3871    \glstextaccessdisplay
3872    {%
3873      \Glsentrytext{#1}%
3874    }%
3875    {#1}%
3876  }

```

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

```

3877  \newcommand*{\GLSaccesstext}[1]{%
3878    \glstextaccessdisplay
3879    {%
3880      \mfirstucMakeUppercase{\glsentrytext{#1}}%
3881    }%
3882    {#1}%
3883  }

```

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

```

3884  \newcommand*{\glsaccessplural}[1]{%
3885    \glspluralaccessdisplay
3886    {%
3887      \glsentryplural{#1}%
3888    }%
3889    {#1}%
3890  }

```

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

```

3891  \newcommand*{\Glsaccessplural}[1]{%
3892    \glspluralaccessdisplay
3893    {%
3894      \Glsentryplural{#1}%

```

```

3895     }%
3896     {#1}%
3897 }

```

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

```

3898 \newcommand*{\GLSaccessplural}[1]{%
3899   \glspluralaccessdisplay
3900   {%
3901     \mfirstucMakeUppercase{\glsentryplural{#1}}%
3902   }%
3903   {#1}%
3904 }

```

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

```

3905 \newcommand*{\glsaccessfirst}[1]{%
3906   \glsfirstaccessdisplay
3907   {%
3908     \glsentryfirst{#1}%
3909   }%
3910   {#1}%
3911 }

```

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

```

3912 \newcommand*{\Glsaccessfirst}[1]{%
3913   \glsfirstaccessdisplay
3914   {%
3915     \Glsentryfirst{#1}%
3916   }%
3917   {#1}%
3918 }

```

\GLSaccessfirst Display the first value (no link and no check for existence) converted to upper case.

```

3919 \newcommand*{\GLSaccessfirst}[1]{%
3920   \glsfirstaccessdisplay
3921   {%
3922     \mfirstucMakeUppercase{\glsentryfirst{#1}}%
3923   }%
3924   {#1}%
3925 }

```

cessfirstplural Display the firstplural value (no link and no check for existence).

```

3926 \newcommand*{\glsaccessfirstplural}[1]{%
3927   \glsfirstpluralaccessdisplay
3928   {%
3929     \glsentryfirstplural{#1}%
3930   }%
3931   {#1}%
3932 }

```


cessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```

3933 \newcommand*\Glsaccessfirstplural}[1]{%
3934 \glfirstpluralaccessdisplay
3935 {%
3936 \Glsentryfirstplural{#1}%
3937 }%
3938 {#1}%
3939 }

```

cessfirstplural Display the firstplural value (no link and no check for existence) converted to upper case.

```

3940 \newcommand*\GLSaccessfirstplural}[1]{%
3941 \glfirstpluralaccessdisplay
3942 {%
3943 \mfirstucMakeUppercase{\Glsentryfirstplural{#1}}%
3944 }%
3945 {#1}%
3946 }

```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```

3947 \newcommand*\glsaccesssymbol}[1]{%
3948 \glssymbolaccessdisplay
3949 {%
3950 \Glsentrysymbol{#1}%
3951 }%
3952 {#1}%
3953 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```

3954 \newcommand*\GLSaccesssymbol}[1]{%
3955 \glssymbolaccessdisplay
3956 {%
3957 \Glsentrysymbol{#1}%
3958 }%
3959 {#1}%
3960 }

```

GLSaccesssymbol Display the symbol value (no link and no check for existence) converted to upper case.

```

3961 \newcommand*\GLSaccesssymbol}[1]{%
3962 \glssymbolaccessdisplay
3963 {%
3964 \mfirstucMakeUppercase{\Glsentrysymbol{#1}}%
3965 }%
3966 {#1}%
3967 }

```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```

3968 \newcommand*{\glsaccesssymbolplural}[1]{%
3969   \glssymbolpluralaccessdisplay
3970   {%
3971     \glsentrysymbolplural{#1}%
3972   }%
3973   {#1}%
3974 }

```

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

```

3975 \newcommand*{\Glsaccesssymbolplural}[1]{%
3976   \glssymbolpluralaccessdisplay
3977   {%
3978     \Glsentrysymbolplural{#1}%
3979   }%
3980   {#1}%
3981 }

```

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

```

3982 \newcommand*{\GLSaccesssymbolplural}[1]{%
3983   \glssymbolpluralaccessdisplay
3984   {%
3985     \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
3986   }%
3987   {#1}%
3988 }

```

`\glsaccessdesc` Display the desc value (no link and no check for existence).

```

3989 \newcommand*{\glsaccessdesc}[1]{%
3990   \glsdescriptionaccessdisplay
3991   {%
3992     \glsentrydesc{#1}%
3993   }%
3994   {#1}%
3995 }

```

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

```

3996 \newcommand*{\Glsaccessdesc}[1]{%
3997   \glsdescriptionaccessdisplay
3998   {%
3999     \Glsentrydesc{#1}%
4000   }%
4001   {#1}%
4002 }

```

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

```

4003 \newcommand*{\GLSaccessdesc}[1]{%

```

```

4004   \glsdescriptionaccessdisplay
4005   {%
4006       \mfirstucMakeUppercase{\glsentrydesc{#1}}%
4007   }%
4008   {#1}%
4009   }

```

`\glsaccessdescplural` Display the descplural value (no link and no check for existence).

```

4010   \newcommand*{\glsaccessdescplural}[1]{%
4011       \glsdescriptionpluralaccessdisplay
4012       {%
4013           \glsentrydescplural{#1}%
4014       }%
4015       {#1}%
4016   }

```

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

```

4017   \newcommand*{\Glsaccessdescplural}[1]{%
4018       \glsdescriptionpluralaccessdisplay
4019       {%
4020           \Glsentrydescplural{#1}%
4021       }%
4022       {#1}%
4023   }

```

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

```

4024   \newcommand*{\GLSaccessdescplural}[1]{%
4025       \glsdescriptionpluralaccessdisplay
4026       {%
4027           \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
4028       }%
4029       {#1}%
4030   }

```

`\glsaccesssshort` Display the short form (no link and no check for existence).

```

4031   \newcommand*{\glsaccesssshort}[1]{%
4032       \glsshortaccessdisplay
4033       {%
4034           \glsentryshort{#1}%
4035       }%
4036       {#1}%
4037   }

```

`\Glsaccesssshort` Display the short form with first letter converted to uppercase (no link and no check for existence).

```

4038   \newcommand*{\Glsaccesssshort}[1]{%
4039       \glsshortaccessdisplay

```

```

4040     {%
4041         \Glsentryshort{#1}%
4042     }%
4043     {#1}%
4044 }

```

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

```

4045 \newcommand*{\GLSaccessshort}[1]{%
4046     \glsshortaccessdisplay
4047     {%
4048         \mfirstucMakeUppercase{\Glsentryshort{#1}}%
4049     }%
4050     {#1}%
4051 }

```

`lsaccessshortpl` Display the short plural form (no link and no check for existence).

```

4052 \newcommand*{\lsaccessshortpl}[1]{%
4053     \glsshortpluralaccessdisplay
4054     {%
4055         \Glsentryshortpl{#1}%
4056     }%
4057     {#1}%
4058 }

```

`lSaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```

4059 \newcommand*{\lSaccessshortpl}[1]{%
4060     \glsshortpluralaccessdisplay
4061     {%
4062         \Glsentryshortpl{#1}%
4063     }%
4064     {#1}%
4065 }

```

`LSaccessshortpl` Display the shortplural value (no link and no check for existence) converted to upper case.

```

4066 \newcommand*{\LSaccessshortpl}[1]{%
4067     \glsshortpluralaccessdisplay
4068     {%
4069         \mfirstucMakeUppercase{\Glsentryshortpl{#1}}%
4070     }%
4071     {#1}%
4072 }

```

`\glsaccesslong` Display the long form (no link and no check for existence).

```

4073 \newcommand*{\glsaccesslong}[1]{%
4074     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
4075 }

```

```

\GLsaccesslong   Display the long form (no link and no check for existence).
4076
4077   \newcommand*{\GLsaccesslong}[1]{%
4078     \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
4079   }

\GLSaccesslong   Display the long value (no link and no check for existence) converted to upper case.
4080   \newcommand*{\GLSaccesslong}[1]{%
4081     \glslongaccessdisplay
4082     {%
4083       \mfirstucMakeUppercase{\Glsentrylong{#1}}%
4084     }%
4085     {#1}%
4086   }

glsaccesslongpl   Display the long plural form (no link and no check for existence).
4087   \newcommand*{\glsaccesslongpl}[1]{%
4088     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
4089   }

Glsaccesslongpl   Display the long plural form (no link and no check for existence).
4090
4091   \newcommand*{\Glsaccesslongpl}[1]{%
4092     \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
4093   }

GLSaccesslongpl   Display the longplural value (no link and no check for existence) converted to upper case.
4094   \newcommand*{\GLSaccesslongpl}[1]{%
4095     \glslongpluralaccessdisplay
4096     {%
4097       \mfirstucMakeUppercase{\Glsentrylongpl{#1}}%
4098     }%
4099     {#1}%
4100   }

      End of if part
4101 }
4102 {
      No accessibility support. Just define these commands to do \glsentry<xxx>

\glsaccessname   Display the name value (no link and no check for existence).
4103   \newcommand*{\glsaccessname}[1]{\Glsentryname{#1}}

\Glsaccessname   Display the name value (no link and no check for existence) with the first letter converted to
upper case.
4104   \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}

```

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

```

4105 \newcommand*{\GLSaccessname}[1]{%
4106 \protect\mfirstucMakeUppercase{\glentryname{#1}}}
```

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

```

4107 \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.

```

4108 \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}
```

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

```

4109 \newcommand*{\GLSaccesstext}[1]{%
4110 \protect\mfirstucMakeUppercase{\glentrytext{#1}}}
```

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

```

4111 \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.

```

4112 \newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}
```

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

```

4113 \newcommand*{\GLSaccessplural}[1]{%
4114 \protect\mfirstucMakeUppercase{\glentryplural{#1}}}
```

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

```

4115 \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.

```

4116 \newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}
```

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

```

4117 \newcommand*{\GLSaccessfirst}[1]{%
4118 \protect\mfirstucMakeUppercase{\glentryfirst{#1}}}
```

`cessfirstplural` Display the firstplural value (no link and no check for existence).

```

4119 \newcommand*{\glsaccessfirstplural}[1]{\glentryfirstplural{#1}}
```

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

```

4120 \newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}
```

`GLSfirstplural` Display the firstplural value (no link and no check for existence). converted to upper case.

```

4121 \newcommand*{\GLSaccessfirstplural}[1]{%
4122 \protect\mfirstucMakeUppercase{\glentryfirstplural{#1}}}
```

`\glsaccesssymbol` Display the symbol value (no link and no check for existence).
4123 `\newcommand*{\glsaccesssymbol}[1]{\glsentrysymbol{#1}}`

`\Glsaccesssymbol` Display the symbol value (no link and no check for existence) with the first letter converted to upper case.
4124 `\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}`

`\GLSaccesssymbol` Display the symbol value (no link and no check for existence). converted to upper case.
4125 `\newcommand*{\GLSaccesssymbol}[1]{%`
4126 `\protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}`

`\glsaccesssymbolplural` Display the symbolplural value (no link and no check for existence).
4127 `\newcommand*{\glsaccesssymbolplural}[1]{\glsentrysymbolplural{#1}}`

`\Glsaccesssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.
4128 `\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}`

`\GLSaccesssymbolplural` Display the symbolplural value (no link and no check for existence). converted to upper case.
4129 `\newcommand*{\GLSaccesssymbolplural}[1]{%`
4130 `\protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}`

`\glsaccessdesc` Display the desc value (no link and no check for existence).
4131 `\newcommand*{\glsaccessdesc}[1]{\glsentrydesc{#1}}`

`\Glsaccessdesc` Display the desc value (no link and no check for existence) with the first letter converted to upper case.
4132 `\newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}`

`\GLSaccessdesc` Display the desc value (no link and no check for existence). converted to upper case.
4133 `\newcommand*{\GLSaccessdesc}[1]{%`
4134 `\protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}`

`\glsaccessdescplural` Display the descplural value (no link and no check for existence).
4135 `\newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}`

`\Glsaccessdescplural` Display the descplural value (no link and no check for existence) with the first letter converted to upper case.
4136 `\newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}`

`\GLSaccessdescplural` Display the descplural value (no link and no check for existence). converted to upper case.
4137 `\newcommand*{\GLSaccessdescplural}[1]{%`
4138 `\protect\mfirstucMakeUppercase{\glsentrydescplural{#1}}}`

`\glsaccessshort` Display the short form (no link and no check for existence).
4139 `\newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}`

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```
4140 \newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}
```

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.

```
4141 \newcommand*{\GLSaccessshort}[1]{%
4142 \protect\mfirstucMakeUppercase{\glsentryshort{#1}}}
```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```
4143 \newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}
```

lSaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```
4144 \newcommand*{\lSaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.

```
4145 \newcommand*{\GLSaccessshortpl}[1]{%
4146 \protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}
```

\glsaccesslong Display the long form (no link and no check for existence).

```
4147 \newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}
```

\Glsaccesslong Display the long form (no link and no check for existence).

```
4148 \newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}
```

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.

```
4149 \newcommand*{\GLSaccesslong}[1]{%
4150 \protect\mfirstucMakeUppercase{\glsentrylong{#1}}}
```

glsaccesslongpl Display the long plural form (no link and no check for existence).

```
4151 \newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}
```

Glsaccesslongpl Display the long plural form (no link and no check for existence).

```
4152 \newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

GLSaccesslongpl Display the longplural value (no link and no check for existence). converted to upper case.

```
4153 \newcommand*{\GLSaccesslongpl}[1]{%
4154 \protect\mfirstucMakeUppercase{\glsentrylongpl{#1}}}
```

End of else part

```
4155 }
```


1.5 Categories

`\glscategory` Add a new storage key that can be used to indicate a category. The default category is general.

```
4156 \glsaddstoragekey{category}{general}{\glscategory}
```

`\glsifcategory` Convenient shortcut to determine if an entry has the given category.

```
4157 \newcommand{\glsifcategory}[4]{%
4158   \ifglsfieldeq{#1}{category}{#2}{#3}{#4}%
4159 }
```

Categories can have attributes.

`categoryattribute`

```
\glssetcategoryattribute{<category>}{<attribute-label>}{<value>}
```

Set (or override if already set) an attribute for the given category.

```
4160 \newcommand*{\glssetcategoryattribute}[3]{%
4161   \csdef{@glxtr@categoryattr@@#1@#2}{#3}%
4162 }
```

`categoryattribute`

```
\glsgetcategoryattribute{<category>}{<attribute-label>}
```

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
4163 \newcommand*{\glsgetcategoryattribute}[2]{%
4164   \csuse{@glxtr@categoryattr@@#1@#2}%
4165 }
```

`categoryattribute`

```
\glshascategoryattribute{<category>}{<attribute-label>}{<true>}{<false>}
```

Tests if the category has the given attribute set.

```
4166 \newcommand*{\glshascategoryattribute}[4]{%
4167   \ifcvoid{@glxtr@categoryattr@@#1@#2}{#4}{#3}%
4168 }
```

`\glssetattribute`

```
\glssetattribute{<entry label>}{<attribute-label>}{<value>}
```

Short cut where the category label is obtained from the entry information.

```
4169 \newcommand*{\glssetattribute}[3]{%
```

```

4170 \glsssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
4171 }

```

`\glsggetattribute` `\glsggetattribute{<entry label>}{<attribute-label>}`

Short cut where the category label is obtained from the entry information.

```

4172 \newcommand*{\glsggetattribute}[2]{%
4173   \glsssetcategoryattribute{\glscategory{#1}}{#2}%
4174 }

```

`\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.

```

4175 \newcommand*{\glshasattribute}[4]{%
4176   \ifglsentryexists{#1}%
4177   {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
4178   {#4}%
4179 }

```

`categoryattribute` `\glssifcategoryattribute{<category>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

True if category has the attribute with the given value.

```

4180 \newcommand{\glssifcategoryattribute}[5]{%
4181   \ifcsundef{@glsxtr@categoryattr@#1@#2}%
4182   {#5}%
4183   {\ifcsstring{@glsxtr@categoryattr@#1@#2}{#3}{#4}{#5}}%
4184 }

```

`\glssifattribute` `\glssifattribute{<entry label>}{<attribute-label>}{<value>}{<true part>}{<false part>}`

Short cut to determine if the given entry has a category with the given attribute set.

```

4185 \newcommand{\glssifattribute}[5]{%
4186   \ifglsentryexists{#1}%
4187   {\glssifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
4188   {#5}%
4189 }

```

Set attributes for the default general category:

```
4190 \glssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
4191 \glssetcategoryattribute{acronym}{regular}{true}
```

`regularcategory` Convenient shortcut to create add the regular attribute.

```
4192 \newcommand*{\glssetregularcategory}[1]{%  
4193   \glssetcategoryattribute{#1}{regular}{true}}%  
4194 }
```

`ifregularcategory` `\glsifregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```
4195 \newcommand{\glsifregularcategory}[3]{%  
4196   \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}}%  
4197 }
```

`ifnotregularcategory` `\glsifnotregularcategory{<category>}{<true part>}{<false part>}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```
4198 \newcommand{\glsifnotregularcategory}[3]{%  
4199   \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}}%  
4200 }
```

`\glsifregular` `\glsifregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to true.

```
4201 \newcommand{\glsifregular}[3]{%  
4202   \glsifregularcategory{\glscategory{#1}}{#2}{#3}}%  
4203 }
```

`\glsifnotregular` `\glsifnotregular{<entry label>}{<true part>}{<false part>}`

Short cut to determine if an entry has a regular attribute set to false.

```
4204 \newcommand{\glsifnotregular}[3]{%  
4205   \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}}%  
4206 }
```

oreachincategory

```
\glsforeachincategory[⟨glossary labels⟩]{⟨category-label⟩}
{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

Iterates through all entries in all the glossaries (or just those listed in *⟨glossary labels⟩*) and does *⟨body⟩* if the category matches *⟨category-label⟩*. The control sequences *⟨glossary-cs⟩* and *⟨label-cs⟩* may be used in *⟨body⟩* to access the glossary label and entry label for the current iteration.

```
4207 \newcommand{\glsforeachincategory}[5][\@glo@types]{%
4208   \forallglossaries[#1]{#3}%
4209   {%
4210     \forlgsentries[#3]{#4}%
4211     {%
4212       \glsifcategory{#4}{#2}{#5}{}%
4213     }%
4214   }%
4215 }
```

achwithattribute

```
\glsforeachwithattribute[⟨glossary labels⟩]{⟨attribute-label⟩}
{⟨attribute-value⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

Iterates through all entries in all the glossaries (or just those listed in *⟨glossary labels⟩*) and does *⟨body⟩* if the category attribute *⟨attribute-label⟩* matches *⟨attribute-value⟩*. The control sequences *⟨glossary-cs⟩* and *⟨label-cs⟩* may be used in *⟨body⟩* to access the glossary label and entry label for the current iteration.

```
4216 \newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
4217   \forallglossaries[#1]{#4}%
4218   {%
4219     \forlgsentries[#4]{#5}%
4220     {%
4221       \glsifattribute{#5}{#2}{#3}{#6}{}%
4222     }%
4223   }%
4224 }
```

If `\newterm` has been defined, redefine it so that it automatically sets the category label to `index` and add `\glstrpostdescription`.

```
4225 \ifdef\newterm
4226 {%
```

`\newterm`

```
4227   \renewcommand*{\newterm}[2][ ]{%
4228     \newglossaryentry{#2}%
4229     {type={index},category=index,name={#2},%
```

```

4230     description={\glxtrpostdescription\nopostdesc},#1}%
4231 }

```

Indexed terms are regular by default.

```

4232 \glsssetcategoryattribute{index}{regular}{true}

```

trpostdescindex

```

4233 \newcommand*{\glxtrpostdescindex}{}

4234 }
4235 {}

```

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.

```

4236 \ifdef\printsymbols
4237 {%

```

glxtrnewsymbol Unlike \newterm, this has a separate argument for the label (since the symbol will likely contain commands).

```

4238 \newcommand*{\glxtrnewsymbol}[3] [] {%
4239     \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
4240 }

```

Symbols are regular by default.

```

4241 \glsssetcategoryattribute{symbol}{regular}{true}

```

rpostdescsymbol

```

4242 \newcommand*{\glxtrpostdescsymbol}{}

4243 }
4244 {}

```

Similar for the numbers option.

```

4245 \ifdef\printnumbers
4246 {%

```

glxtrnewnumber

```

4247 \ifdef\printnumbers
4248 \newcommand*{\glxtrnewnumber}[3] [] {%
4249     \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
4250 }

```

Numbers are regular by default.

```

4251 \glsssetcategoryattribute{number}{regular}{true}

```

rpostdescnumber

```

4252 \newcommand*{\glxtrpostdescnumber}{}

```

```
4253 }
4254 {}
```

`\glstrsetcategory` Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
4255 \newcommand*{\glstrsetcategory}[2]{%
4256   \@for\@glstr@label:=#1\do
4257   {%
4258     \glsfieldxdef{\@glstr@label}{category}{#2}%
4259   }%
4260 }
```

`\glstrcategoryforall` Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```
4261 \newcommand*{\glstrcategoryforall}[2]{%
4262   \forallglossaries[#1]{\@glstr@type}{%
4263     \forallsentries[\@glstr@type]{\@glstr@label}%
4264     {%
4265       \glsfieldxdef{\@glstr@label}{category}{#2}%
4266     }%
4267   }%
4268 }
```

`\glstrfieldtitlecase` `\glstrfieldtitlecase{\langle label \rangle}{\langle field \rangle}`

Apply title casing to the contents of the given field.

```
4269 \newcommand*{\glstrfieldtitlecase}[2]{%
4270   \expandafter\glstrfieldtitlecasesecs\expandafter
4271   {\csname glo@glsdetoklabel{#1}@#2\endcsname}%
4272 }
```

`\glstrfieldtitlecasesecs` The command used by `\glstrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```
4273 \newcommand*{\glstrfieldtitlecasesecs}[1]{\xcapitalisewords{#1}}
```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

`\glossentrydesc` If the `glossdesc` attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```
4274 \@ifpackageloaded{glossaries-accsupp}
4275 {
4276   \renewcommand*{\glossentrydesc}[1]{%
4277     \glsdoifexistsorwarn{#1}%
4278     {%
4279       \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```

4280 \glshasattribute{#1}{glossdescfont}%
4281 {%
4282 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
4283 \ifcsdef{\@glxtr@attrval}%
4284 {%
4285 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
4286 }%
4287 {%
4288 \GlossariesExtraWarning{Unknown control sequence name
4289 '\@glxtr@attrval' supplied in glossdescfont attribute
4290 for entry '#1'. Ignoring}%
4291 \let\@glxtr@glossdescfont\@firstofone
4292 }%
4293 }%
4294 {\let\@glxtr@glossdescfont\@firstofone}%
4295 \glusifattribute{#1}{glossdesc}{firstuc}%
4296 {%
4297 \@glxtr@glossdescfont{\Glsaccessdesc{#1}}%
4298 }%
4299 {%
4300 \glusifattribute{#1}{glossdesc}{title}%
4301 {%
4302 \@glxtr@do@titlecaps@warn
4303 \glsdescriptionaccessdisplay
4304 {%
4305 \@glxtr@glossdescfont{\glxtrfieldtitlecase{#1}{desc}}%
4306 }%
4307 {#1}%
4308 }%
4309 {%
4310 \@glxtr@glossdescfont{\glaccessdesc{#1}}%
4311 }%
4312 }%
4313 }%
4314 }
4315 }
4316 {
4317 \renewcommand*\glossentrydesc[1]{%
4318 \glsdoifexistsorwarn{#1}%
4319 {%
4320 \glissetabbrvfmt{\glscategory{#1}}%
4321 \glshasattribute{#1}{glossdescfont}%
4322 {%
4323 \edef\@glxtr@attrval{\glsggetattribute{#1}{glossdescfont}}%
4324 \ifcsdef{\@glxtr@attrval}%
4325 {%
4326 \letcs{\@glxtr@glossdescfont}{\@glxtr@attrval}%
4327 }%

```

```

4328      {%
4329      \GlossariesExtraWarning{Unknown control sequence name
4330      '\@glsxtr@attrval' supplied in glossdescfont attribute
4331      for entry '#1'. Ignoring}%
4332      \let\@glsxtr@glossdescfont\@firstofone
4333      }%
4334    }%
4335    {\let\@glsxtr@glossdescfont\@firstofone}%
4336    \glsifattribute{#1}{glossdesc}{firstuc}%
4337    {%
4338      \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
4339    }%
4340    {%
4341      \glsifattribute{#1}{glossdesc}{title}%
4342      {%
4343        \@glsxtr@do@titlecaps@warn
4344        \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
4345      }%
4346      {%
4347        \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
4348      }%
4349    }%
4350  }%
4351 }
4352 }

```

`\glossentryname` If the glossname attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```

4353 \ifpackageloaded{glossaries-accsupp}
4354 {
4355   \renewcommand*{\glossentryname}[1]{%
4356     \@glsdoifexistsorwarn{#1}%
4357     {%
4358       \glssetabbrvfmt{\glscategory{#1}}%

```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```

4359   \glshasattribute{#1}{glossnamefont}%
4360   {%
4361     \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
4362     \ifcsdef{\@glsxtr@attrval}%
4363     {%
4364       \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
4365     }%
4366     {%
4367       \GlossariesExtraWarning{Unknown control sequence name
4368       '\@glsxtr@attrval' supplied in glossnamefont attribute
4369       for entry '#1'. Reverting to default \string\glsnamefont}%
4370       \let\@glsxtr@glossnamefont\glsnamefont
4371     }%
4372   }%

```



```

4373     {\let\@glsxtr@glossnamefont\glsnamefont}%
4374     \glsifattribute{#1}{glossname}{firstuc}%
4375     {%
4376         \glsnameaccessdisplay
4377         {%
4378             \@glsxtr@glossnamefont{\Glsentryname{#1}}%
4379         }%
4380     {#1}%
4381 }%
4382 {%
4383     \glsifattribute{#1}{glossname}{title}%
4384     {%
4385         \@glsxtr@do@titlecaps@warn
4386         \glsnameaccessdisplay
4387         {%
4388             \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
4389         }%
4390     {#1}%
4391 }%
4392 {%
4393     \glsifattribute{#1}{glossname}{uc}%
4394     {%
4395         \glsnameaccessdisplay
4396         {%

```

Hide the label from the upper-casing command.

```

4397         \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
4398         \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
4399     }%
4400 {#1}%
4401 }%
4402 {%
4403     \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
4404     \glsnameaccessdisplay
4405     {%
4406         \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
4407     }%
4408 {#1}%
4409 }%
4410 }%
4411 }%

```

Do post-name hook:

```

4412     \glsxtrpostnamehook{#1}%
4413 }%
4414 }
4415 }
4416 {
4417     \renewcommand*{\glossentryname}[1]{%
4418         \@glsdoifexistsorwarn{#1}%

```

```

4419  {%
4420      \glssetabbrvfmt{\glscategory{#1}}%
4421      \glshasattribute{#1}{glossnamefont}%
4422      {%
4423          \edef\@glstr@attrval{\glsgetattribute{#1}{glossnamefont}}%
4424          \ifcsdef{\@glstr@attrval}%
4425              {%
4426                  \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
4427              }%
4428              {%
4429                  \GlossariesExtraWarning{Unknown control sequence name
4430                      '\@glstr@attrval' supplied in glossnamefont attribute
4431                      for entry '#1'. Reverting to default \string\glsnamefont}%
4432                  \let\@glstr@glossnamefont\glsnamefont
4433              }%
4434          }%
4435          {\let\@glstr@glossnamefont\glsnamefont}%
4436          \glsifattribute{#1}{glossname}{firstuc}%
4437          {%
4438              \@glstr@glossnamefont{\Glsentryname{#1}}%
4439          }%
4440          {%
4441              \glsifattribute{#1}{glossname}{title}%
4442              {%
4443                  \@glstr@do@titlecaps@warn
4444                  \@glstr@glossnamefont{\glstrfieldtitlecase{#1}{name}}%
4445              }%
4446              {%
4447                  \glsifattribute{#1}{glossname}{uc}%
4448                  {%

```

Hide the label from the upper-casing command.

```

4449          \letcs{\glo@name}{\glo\glsdetoklabel{#1}@name}%
4450          \@glstr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
4451      }%
4452  {%

```

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.

```

4453          \letcs{\glo@name}{\glo\glsdetoklabel{#1}@name}%
4454          \expandafter\@glstr@glossnamefont\expandafter{\glo@name}%
4455      }%
4456  }%
4457 }%

```

Do post-name hook.

```

4458      \glstrpostnamehook{#1}%
4459  }%
4460 }
4461 }

```

`\Glossentryname` Redefine to set the abbreviation format and accessibility support.

```
4462 \ifpackageloaded{glossaries-accsupp}
4463 {
4464   \renewcommand*{\Glossentryname}[1]{%
4465     \@glsdoifexistsorwarn{#1}%
4466     {%
4467       \glsetabbrvfmt{\glscategory{#1}}%
4468       \glshasattribute{#1}{glossnamefont}%
4469       {%
4470         \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
4471         \ifcsdef{\@glstr@attrval}%
4472         {%
4473           \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
4474           }%
4475           {%
4476             \GlossariesExtraWarning{Unknown control sequence name
4477               '\@glstr@attrval' supplied in glossnamefont attribute
4478               for entry '#1'. Reverting to default \string\glnamefont}%
4479             \let\@glstr@glossnamefont\glnamefont
4480             }%
4481             }%
4482             {\let\@glstr@glossnamefont\glnamefont}%
4483             \glnameaccessdisplay
4484             {%
4485               \@glstr@glossnamefont{\Glsentryname{#1}}%
4486             }%
4487             {#1}%
4488             \glstrpostnamehook{#1}%
4489             }%
4490           }
4491 }
4492 {
4493   \renewcommand*{\Glossentryname}[1]{%
4494     \@glsdoifexistsorwarn{#1}%
4495     {%
4496       \glsetabbrvfmt{\glscategory{#1}}%
4497       \glshasattribute{#1}{glossnamefont}%
4498       {%
4499         \edef\@glstr@attrval{\glsetattribute{#1}{glossnamefont}}%
4500         \ifcsdef{\@glstr@attrval}%
4501         {%
4502           \letcs{\@glstr@glossnamefont}{\@glstr@attrval}%
4503           }%
4504           {%
4505             \GlossariesExtraWarning{Unknown control sequence name
4506               '\@glstr@attrval' supplied in glossnamefont attribute
```

```

4507         for entry ‘#1’. Reverting to default \string\glsnamefont}%
4508         \let\@glsxtr@glossnamefont\glsnamefont
4509     }%
4510 }%
4511 {\let\@glsxtr@glossnamefont\glsnamefont}%
4512 \@glsxtr@glossnamefont{\Glsentryname{#1}}}%

```

Do post-name hook:

```

4513     \glsxtrpostnamehook{#1}%
4514 }%
4515 }
4516 }

```

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`xtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```

4517 \newcommand*{\glsxtrpostnamehook}[1]{%
4518   \def\@glsnumberformat{glsnumberformat}%
4519   \glsxtrdoautoindexname{#1}{indexname}%

```

Allow categories to hook in here.

```

4520   \csuse{glsxtrpostname\glscategory{\glscurrententrylabel}}%
4521 }

```

`format@override` Determines if the format key should override the indexing attribute value.

```

4522 \newif\ifglsxtr@format@override
4523 \@glsxtr@format@overridefalse

```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```

4524 \@ifpackageloaded{hyperref}
4525 {

```

If `hyperref`'s `hyperindex` option is on, then `hyperref` will automatically add `\hyperpage`, so don't add it.

```

4526   \ifHy@hyperindex
4527     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
4528       \@glsxtr@format@override true
4529       \appto\theindex{\let\glshypernumber\@firstofone}%
4530     }
4531   \else
4532     \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
4533       \@glsxtr@format@override true
4534       \appto\theindex{\let\glshypernumber\hyperpage}%
4535     }
4536   \fi

```

```

4537 }
4538 {
4539   \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
4540     \@glstr@format@overridetrue
4541   }
4542 }
4543 \onlypreamble\GlsXtrEnableIndexFormatOverride

```

doautoindexname

```

4544 \newcommand*{\glstrdoautoindexname}[2]{%
4545   \glshasattribute{#1}{#2}%
4546   {%

```

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.

```

4547   \@glstr@autoindex@setname{#1}%

```

If the attribute value is simply “true” don't add an encap, otherwise use the value as the encap.

```

4548   \protected@edef\@glstr@attrval{\glsggetattribute{#1}{#2}}%
4549   \if@glstr@format@override
4550     \ifdefstring{\@glstr@numberformat}{\glstrnumberformat}{}%
4551     {\let\@glstr@attrval\@glstrnumberformat}%
4552   \fi
4553   \ifdefstring{\@glstr@attrval}{true}%
4554   {%
4555     {\eappto\@glo@name{\@glstr@autoindex@encap\@glstr@attrval}}%
4556     \expandafter\index\expandafter{\@glo@name}%
4557   }%
4558   {}%
4559 }

```

toindex@setname Assign \@glo@name for use with indexname attribute.

```

4560 \newcommand*{\@glstr@autoindex@setname}[1]{%
4561   \def\@glo@name{\string\glstryname{#1}}%
4562   \glstryfield{\@glo@sort}{#1}{sort}%
4563   \@glstr@checkmkidxchars\@glo@sort
4564   \@glstr@autoindex@doextra@esc\@glo@sort
4565   \epreto\@glo@name{\@glo@sort\@glstr@autoindex@at}%
4566 }

```

dex@doextra@esc

```

4567 \newcommand*{\@glstr@autoindex@doextra@esc}[1]{%

```

Escape the escape character unless it has already been escaped.

```

4568   \ifx\@glstr@autoindex@esc\@glstr@quotechar
4569   \else
4570     \def\@glstr@checkedmkidx{%
4571       \edef\@glstr@checkspch{%
4572         \noexpand\@glstr@autoindex@escquote\expandonce{#1}%
4573         \noexpand\@empty\@glstr@autoindex@esc\noexpand\@nnil

```

```

4574      \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
4575      \@glsxtr@checkspch
4576      \let#1\@gls@checkedmkidx\relax
4577 \fi

```

Escape actual character unless it has already been escaped.

```

4578 \ifx\@glsxtr@autoindex@at\@gls@actualchar
4579 \else
4580   \def\@gls@checkedmkidx{}%
4581   \edef\@glsxtr@checkspch{%
4582     \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
4583     \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
4584     \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
4585   \@glsxtr@checkspch
4586   \let#1\@gls@checkedmkidx\relax
4587 \fi

```

Escape level character unless it has already been escaped.

```

4588 \ifx\@glsxtr@autoindex@level\@gls@levelchar
4589 \else
4590   \def\@gls@checkedmkidx{}%
4591   \edef\@glsxtr@checkspch{%
4592     \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
4593     \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
4594     \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
4595   \@glsxtr@checkspch
4596   \let#1\@gls@checkedmkidx\relax
4597 \fi

```

Escape encap character unless it has already been escaped.

```

4598 \ifx\@glsxtr@autoindex@encap\@gls@encapchar
4599 \else
4600   \def\@gls@checkedmkidx{}%
4601   \edef\@glsxtr@checkspch{%
4602     \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
4603     \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
4604     \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
4605   \@glsxtr@checkspch
4606   \let#1\@gls@checkedmkidx\relax
4607 \fi
4608 }

```

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`.

```
4609 \newcommand*{\@glsxtr@autoindex@at}{}

```

`\trSetActualChar` Set the actual character.

```
4610 \newcommand*{\GlsXtrSetActualChar}[1]{%

```

```

4611 \gdef\@glsxtr@autoindex@at{#1}%
4612 \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
4613   \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
4614 }%
4615 }
4616 \@onlypreamble\GlsXtrSetActualChar
4617 \makeatother
4618 \GlsXtrSetActualChar{@}
4619 \makeatletter

```

autoindex@encap Encap character for use with `\index`.

```
4620 \newcommand*{\@glsxtr@autoindex@encap}{}
```

XtrSetEncapChar Set the encap character.

```

4621 \newcommand*{\GlsXtrSetEncapChar}[1]{%
4622   \gdef\@glsxtr@autoindex@encap{#1}%
4623   \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
4624     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
4625   }%
4626 }
4627 \GlsXtrSetEncapChar{||}
4628 \@onlypreamble\GlsXtrSetEncapChar

```

autoindex@level Level character for use with `\index`.

```
4629 \newcommand*{\@glsxtr@autoindex@level}{}
```

XtrSetLevelChar Set the encap character.

```

4630 \newcommand*{\GlsXtrSetLevelChar}[1]{%
4631   \gdef\@glsxtr@autoindex@level{#1}%
4632   \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%
4633     \@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
4634   }%
4635 }
4636 \GlsXtrSetLevelChar{!}
4637 \@onlypreamble\GlsXtrSetLevelChar

```

r@autoindex@esc Escape character for use with `\index`.

```
4638 \newcommand*{\@glsxtr@autoindex@esc}{"
```

lsXtrSetEscChar Set the escape character.

```

4639 \newcommand*{\GlsXtrSetEscChar}[1]{%
4640   \gdef\@glsxtr@autoindex@esc{#1}%
4641   \def\@glsxtr@autoindex@escquote##1#1##2#1##3\@glsxtr@endescspch{%
4642     @@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
4643   }%
4644 }
4645 \GlsXtrSetEscChar{"}
4646 \@onlypreamble\GlsXtrSetEscChar

```

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:

```
4647 \ifdef\actualchar
4648 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
4649 {}
```

Quote character \quotechar:

```
4650 \ifdef\quotechar
4651 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
4652 {}
```

Level character \levelchar:

```
4653 \ifdef\levelchar
4654 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
4655 {}
```

Encap character \encapchar:

```
4656 \ifdef\encapchar
4657 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
4658 {}
```

leto@endescspch

```
4659 \def\@glxstr@gobbleto@endescspch#1\@glxstr@endescspch{}
```

toindex@esc@spch

```
\@glxstr@autoindex@escspch{<char>}{<cs>}{<pre>}{<mid>}{<post>}
```

```
4660 \newcommand*{\@glxstr@autoindex@escspch}[5]{%
4661   \@glstmpb=\expandafter{\@glsc@checkedmkidx}%
4662   \toks@={#3}%
4663   \ifx\@nnil#3\relax
4664     \def\@glxstr@checkspch{\@glxstr@gobbleto@endescspch#5\@glxstr@endescspch}%
4665   \else
4666     \ifx\@nnil#4\relax
4667       \edef\@glsc@checkedmkidx{\the\@glstmpb\the\toks@}%
4668       \def\@glxstr@checkspch{\@glxstr@gobbleto@endescspch
4669         #4#5\@glxstr@endescspch}%
4670     \else
4671       \edef\@glsc@checkedmkidx{\the\@glstmpb\the\toks@
4672         \@glxstr@autoindex@esc#1}%
4673       \def\@glxstr@checkspch{#2#5#1\@nnil#1\@glxstr@endescspch}%
4674     \fi
4675   \fi
4676   \@glxstr@checkspch
4677 }
```

\Glossentrydesc Redefine to set the abbreviation format and accessibility support.

```
4678 \renewcommand*{\Glossentrydesc}[1]{%
4679   \glsoifexistsorwarn{#1}%
```



```

4680  {%
4681    \glssetabbrvfmt{\glscategory{#1}}%
4682    \Glsaccessdesc{#1}%
4683  }%
4684 }

```

`\glossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```

4685 \renewcommand*{\glossentrysymbol}[1]{%
4686   \glsdoifexistsorwarn{#1}%
4687   {%
4688     \glssetabbrvfmt{\glscategory{#1}}%
4689     \glsaccesssymbol{#1}%
4690   }%
4691 }

```

`\Glossentrysymbol` Redefine to set the abbreviation format and accessibility support.

```

4692 \renewcommand*{\Glossentrysymbol}[1]{%
4693   \glsdoifexistsorwarn{#1}%
4694   {%
4695     \glssetabbrvfmt{\glscategory{#1}}%
4696     \Glsaccesssymbol{#1}%
4697   }%
4698 }

```

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.

```

4699 \newcommand*{\GlsXtrEnableInitialTagging}{%
4700   \@ifstar\s@glxtr@enabletagging\@glxtr@enabletagging
4701 }
4702 \@onlypreamble\GlsXtrEnableInitialTagging

```

`\r@enabletagging` Starred version undefines command.

```

4703 \newcommand*{\s@glxtr@enabletagging}[2]{%
4704   \undef#2%
4705   \@glxtr@enabletagging{#1}{#2}%
4706 }

```

`\r@enabletagging` Internal command.

```

4707 \newcommand*{\@glxtr@enabletagging}[2]{%
  Set attributes for categories given in the first argument.
4708   \@for\@glxtr@cat:=#1\do
4709   {%
4710     \ifdefempty\@glxtr@cat
4711     {}%
4712     {\glssetcategoryattribute{\@glxtr@cat}{tagging}{true}}%

```

```

4713 }%
4714 \newrobustcmd*#2[1]{##1}%
4715 \def\@glsxtr@taggingcs{#2}%
4716 \renewcommand*\@glsxtr@activate@initialtagging{%
4717   \let#2\@glsxtr@tag
4718 }%
4719 \ifundef\@gls@preglossaryhook
4720 {\GlossariesExtraWarning{Initial tagging requires at least
4721   glossaries.sty v4.19 to work correctly}}%
4722 {}%
4723 }

```

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

```

4724 \ifundef\mfu@checkword@do
4725 {
4726   \newcommand*\mfu@checkword@do[1]{%
4727     \ifdefstring{\mfu@checkword@arg}{#1}%
4728     {%
4729       \let\@mfu@domakefirstuc\@firstofone
4730       \listbreak
4731     }%
4732   }%
4733 }

```

`\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.

```

4734 \ifundef\mfu@checkword
4735 {
4736   \newcommand{\@glsxtr@do@titlecaps@warn}{%
4737     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
4738       support not available}%

```

One warning should suffice.

```

4739     \let\@glsxtr@do@titlecaps@warn\relax
4740   }
4741 }
4742 {
4743   \renewcommand*\mfu@checkword[1]{%
4744     \def\mfu@checkword@arg{#1}%
4745     \let\@mfu@domakefirstuc\makefirstuc
4746     \forlistloop\mfu@checkword@do\@mfu@nocaplist
4747   }
4748 }
4749 }
4750 {}% no patch required

```

`@titlecaps@warn` Do warning if title case not supported.

4751 `\newcommand*{\@glstr@do@titlecaps@warn}{}`

`@initialtagging` Used in `\printglossary` but at least v4.19 of glossaries required.

4752 `\newcommand*\@glstr@activate@initialtagging{}`

`\@glstr@tag` Definition of tagging command when used in glossary.

4753 `\newrobustcmd*{\@glstr@tag}[1]{%`
4754 `\gl@ifattribute{\glscurrententrylabel}{tagging}{true}%`
4755 `{\glstrtagfont{#1}}{#1}%`
4756 `}`

`\glstrtagfont` Used in the glossary.

4757 `\newcommand*\glstrtagfont[1]{\underline{#1}}`

`preglossaryhook` This macro was introduced in glossaries version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.

4758 `\ifdef\@glspreglossaryhook`
4759 `{`
4760 `\renewcommand*\@glspreglossaryhook{%`
4761 `\glstr@activate@initialtagging`

Since the glossaries are automatically scoped, `\@glstr@org@postdescription` shouldn't already be defined, but check anyway just as a precautionary measure.

4762 `\ifundef\@glstr@org@postdescription`
4763 `{%`
4764 `\let\@glstr@org@postdescription\glspostdescription`
4765 `\renewcommand*\glspostdescription{%`
4766 `\ifglseentryexists{\glscurrententrylabel}%`
4767 `{%`
4768 `\glstrpostdescription`
4769 `\@glstr@org@postdescription`
4770 `}%`
4771 `{}%`
4772 `}%`
4773 `}%`
4774 `{}%`

Enable the options used by `\@glstrp`:

4775 `\glossxtrsetpopts`
4776 `}%`
4777 `}`
4778 `{}`

`postdescription` This command will only be used if `\@glspreglossaryhook` 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.

```

4779 \newcommand*{\glxtrpostdescription}{%
4780   \csuse{glxtrpostdesc\glscategory{\glscurrententrylabel}}%
4781 }

postdescgeneral
4782 \newcommand*{\glxtrpostdescgeneral}{}

xtrpostdescterm
4783 \newcommand*{\glxtrpostdescterm}{}

postdescacronym
4784 \newcommand*{\glxtrpostdescacronym}{}

escabbreviation
4785 \newcommand*{\glxtrpostdescabbreviation}{}

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.
4786 \renewcommand*{\glspostlinkhook}{%
4787   \ifglseentryexists{\glslabel}{\glxtrpostlinkhook}{}%
4788 }

xtrpostlinkhook  The entry label should already be stored in \glslabel by \@gls@link.
4789 \newcommand*{\glxtrpostlinkhook}{%
4790   \glxtrdiscardperiod{\glslabel}%
4791   {\glxtrpostlinkendsentence}%
4792   {\glxtrpostlink}%
4793 }

\glxtrpostlink
4794 \newcommand*{\glxtrpostlink}{%
4795   \csuse{glxtrpostlink\glscategory{\glslabel}}%
4796 }

linkendsentence  Done by \glxtrpostlinkhook if a full stop is discarded.
4797 \newcommand*{\glxtrpostlinkendsentence}{%
4798   \ifcsdef{glxtrpostlink\glscategory{\glslabel}}
4799   {%
4800     \csuse{glxtrpostlink\glscategory{\glslabel}}%
      Put the full stop back.
4801     .\spacefactor\sfcode'\. \relax
4802   }%
4803   {%

```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
4804 \spacefactor\sfcode'\. \relax
4805 }%
4806 }
```

addDescOnFirstUse 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.

```
4807 \newcommand*{\glxtrpostlinkAddDescOnFirstUse}{%
4808 \glxtrifwasfirstuse{\space(\glssaccessdesc{\glslabel})}{}%
4809 }
```

addSymbolOnFirstUse 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.

```
4810 \newcommand*{\glxtrpostlinkAddSymbolOnFirstUse}{%
4811 \glxtrifwasfirstuse
4812 {%
4813 \ifglshassymbol{\glslabel}{\space(\glssaccesssymbol{\glslabel})}{}%
4814 }%
4815 {}%
4816 }
```

discardperiod 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).

```
4817 \newcommand*{\glxtrdiscardperiod}[3]{%
4818 \glxtrifwasfirstuse
4819 {%
4820 \glusifattribute{#1}{retainfirstuseperiod}{true}%
4821 {#3}%
4822 {%
4823 \glusifattribute{#1}{discardperiod}{true}%
4824 {%
4825 \glusifplural
4826 {%
4827 \glusifattribute{#1}{pluraldiscardperiod}{true}%
4828 {\glxtrifperiod{#2}{#3}}%
4829 {#3}%
4830 }%
4831 {%
4832 \glxtrifperiod{#2}{#3}%
4833 }%
4834 }%
4835 {#3}%
4836 }%
4837 }%
4838 {%
```

```

4839 \glsifattribute{#1}{discardperiod}{true}%
4840 {%
4841   \glsifplural
4842   {%
4843     \glsifattribute{#1}{pluraldiscardperiod}{true}%
4844     {\glsxtrifperiod{#2}{#3}}%
4845     {#3}%
4846   }%
4847   {%
4848     \glsxtrifperiod{#2}{#3}%
4849   }%
4850 }%
4851 {#3}%
4852 }%
4853 }

```

`\glsxtrifperiod` Make a convenient user command to check if the next character is a full stop (period). Works like `\@ifstar` but uses `\new@ifnextchar` rather than `\@ifnextchar`

```

4854 \newcommand*{\glsxtrifperiod}[1]{\new@ifnextchar.\@firstoftwo{#1}}

```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

`\glsxtr@punclist` List of characters identified as punctuation marks. (Be careful of babel shorthands!) This doesn't allow for punctuation marks made up from multiple characters (such as ' ').

```

4855 \newcommand*{\glsxtr@punclist}{.,:;?!}

```

`punctuationmark` Add character to punctuation list.

```

4856 \newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}

```

`unctuationmarks` Reset the punctuation list.

```

4857 \newcommand*{\glsxtrsetpunctuationmarks}[1]{\def\glsxtr@punclist{#1}}

```

`\glsxtrifpunc` `\glsxtrifnextpunc{<true part>}{<>false part>}`

Test if this is followed by a punctuation mark. (Adapted from `\new@ifnextchar`.)

```

4858 \newcommand*{\glsxtrifnextpunc}[2]{%
4859   \def\reserved@a{#1}%
4860   \def\reserved@b{#2}%
4861   \futurelet\@glsxpunc@token\glsxtr@ifnextpunc
4862 }

```

`sxtr@ifnextpunc`

```

4863 \newcommand*{\glsxtr@ifnextpunc}{%
4864   \glsxtr@ifpunctoken{\@glsxpunc@token}{\let\reserved@b\reserved@a}{}%
4865   \reserved@b
4866 }

```

```

xtr@ifpunctoken  Test if the token given in the first argument is in the punctuation list.
4867 \newcommand*{\glxtr@ifpunctoken}[1]{%
4868   \expandafter\@glxtr@ifpunctoken\expandafter#1\glxtr@punclist\@nnil
4869 }

```

```

xtr@ifpunctoken
4870 \def\@glxtr@ifpunctoken#1#2{%
4871   \let\reserved@d=#2%
4872   \ifx\reserved@d\@nnil
4873     \let\glxtr@next\@glxtr@notfoundinlist
4874   \else
4875     \ifx#1\reserved@d
4876       \let\glxtr@next\@glxtr@foundinlist
4877     \else
4878       \let\glxtr@next\@glxtr@ifpunctoken
4879     \fi
4880   \fi
4881   \glxtr@next#1%
4882 }

```

```

xtr@foundinlist
4883 \def\@glxtr@foundinlist#1\@nnil{\@firstoftwo}

```

```

@notfoundinlist
4884 \def\@glxtr@notfoundinlist#1{\@secondoftwo}

```

```

glxtrdopostpunc \glxtrdopostpunc{<code>}

```

If this is followed by a punctuation character, do *<code>* after the character otherwise do *<code>* before whatever comes next.

```

4885 \newcommand{\glxtrdopostpunc}[1]{%
4886   \glxtr@ifnextpunc{\@glxtr@swaptwo{#1}}{#1}%
4887 }

```

```

@glxtr@swaptwo
4888 \newcommand{\@glxtr@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.

```

4889 \define@key{glsxtrabbrv}{category}{%
4890 \edef\glscategorylabel{#1}%
4891 \ifcsdef{@glsabbrv@current@#1}%
4892 {%

```

Warning should already have been issued.

```

4893 \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
4894 \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
4895 \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
4896 \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
4897 }%
4898 }%
4899 }

```

Save the short plural form. This may be needed before the entry is defined.

```

4900 \define@key{glsxtrabbrv}{shortplural}{%
4901 \def\@gls@shortpl{#1}%
4902 }

```

Similarly for the long plural form.

```

4903 \define@key{glsxtrabbrv}{longplural}{%
4904 \def\@gls@longpl{#1}%
4905 }

```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```

4906 \newtoks\glsshortpltok

```

\glslongpltok

```

4907 \newtoks\glslongpltok

```

glsxtr@insertdots Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to \newabbreviation. Note that explicitly using the short or shortplural keys will override this.

```

4908 \newcommand*{\@glsxtr@insertdots}[2]{%
4909 \def#1{%
4910 \@glsxtr@insert@dots#1#2\@nnil
4911 }

```

glsxtr@insert@dots

```

4912 \newcommand*{\@glsxtr@insert@dots}[2]{%
4913 \ifx\@nnil#2\relax
4914 \let\@glsxtr@insert@dots@next\@gobble
4915 \else
4916 \ifx\relax#2\relax

```



```

4917 \else
4918     \appto#1{#2.}%
4919 \fi
4920 \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
4921 \fi
4922 \@glsxtr@insert@dots@next#1%
4923 }

```

newabbreviation Define a new generic abbreviation.

```

4924 \newcommand*{\newabbreviation}[4] [] {%
4925     \glsxtr@newabbreviation{#1}{#2}{#3}{#4}%
4926 }

```

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.)

```

4927 \newcommand*{\glsxtr@newabbreviation}[4] {%
4928     \glskeylisttok{#1}%
4929     \glslabeltok{#2}%
4930     \glsshorttok{#3}%
4931     \glslongtok{#4}%

```

Get the category.

```

4932 \def\glscategorylabel{abbreviation}%
4933 \glsxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
4934 \setkeys*{glsxtrabbrv}{shortplural,longplural}{#1}%

```

Set the default long plural

```

4935 \def\@gls@longpl{#4\glspluralsuffix}%

```

Has the insertdots attribute been set?

```

4936 \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
4937 {%
4938     \@glsxtr@insertdots\@gls@short{#3}%
4939     \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
4940     \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
4941     {%
4942         \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
4943             '\abbrvpluralsuffix}%
4944     }%
4945     {%
4946         \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
4947         {%
4948             \let\@gls@shortpl\@gls@short
4949         }%
4950         {%
4951             \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
4952                 \abbrvpluralsuffix}%
4953         }%
4954     }%
4955 }%
4956 {%

```

insertdots not true.

```
4957 \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
4958 {%
4959 \def\@gls@shortpl{#3'\abbrvpluralsuffix}%
4960 }%
4961 {%
4962 \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
4963 {%
4964 \def\@gls@shortpl{#3}%
4965 }%
4966 {%
4967 \def\@gls@shortpl{#3\abbrvpluralsuffix}%
4968 }%
4969 }%
4970 }%
```

Hook for further customisation if required:

```
4971 \glsxtrnewabbrevpresetkeyhook{#1}{#2}{#3}%
```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary.

```
4972 \setkeys*{glsxtrabbrv}[category]{#1}%
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
4973 \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
4974 \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Do any extra setup provided by hook:

```
4975 \newabbreviationhook
```

Define this entry:

```
4976 \protected@edef\@do@newglossaryentry{%
4977 \noexpand\newglossaryentry{\the\glslabeltok}%
4978 {%
4979 type=\glsxtrabbrvtype,%
4980 category=abbreviation,%
4981 short={\the\glsshorttok},%
4982 shortplural={\the\glsshortpltok},%
4983 long={\the\glslongtok},%
4984 longplural={\the\glslongpltok},%
4985 name={\the\glsshorttok},%
4986 \CustomAbbreviationFields,%
4987 \the\glskeylisttok
4988 }%
4989 }%
4990 \@do@newglossaryentry
4991 \GlsXtrPostNewAbbreviation
4992 }
```

evpresetkeyhook Hook for extra stuff in \newabbreviation

```
4993 \newcommand*{\glsxtrnewabbrevpresetkeyhook}[3]{}
```

NewAbbreviation Hook used by abbreviation styles.

```
4994 \newcommand*{\GlsXtrPostNewAbbreviation}{}
```

bbreviationhook Hook for use with `\newabbreviation`.

```
4995 \newcommand*{\newabbreviationhook}{}%
```

reviationFields

```
4996 \newcommand*{\CustomAbbreviationFields}{}%
```

lsxtrfullformat Full format without case change.

```
4997 \newcommand*{\glsxtrfullformat}[2]{%
4998   \glsfirstlongfont{\glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
4999   (\protect\glsfirstabbrvfont{\glsaccessshort{#1}})%
5000 }
```

lsxtrfullformat Full format with case change.

```
5001 \newcommand*{\Glsxtrfullformat}[2]{%
5002   \glsfirstlongfont{\Glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
5003   (\protect\glsfirstabbrvfont{\Glsaccessshort{#1}})%
5004 }
```

xtrfullplformat Plural full format without case change.

```
5005 \newcommand*{\glsxtrfullplformat}[2]{%
5006   \glsfirstlongfont{\glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
5007   (\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}})%
5008 }
```

xtrfullplformat Plural full format with case change.

```
5009 \newcommand*{\Glsxtrfullplformat}[2]{%
5010   \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
5011   (\protect\glsfirstabbrvfont{\Glsaccessshortpl{#1}})%
5012 }
```

\glsxtrfullsep Separator used by full format is a space by default. The argument is the entry's label.

```
5013 \newcommand*{\glsxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with `\glsentryfull` (for example, first use suppresses the long form or uses a footnote).

inlinefullformat Full format without case change.

```
5014 \newcommand*{\glsxtrininlinefullformat}{\glsxtrfullformat}
```

inlinefullformat Full format with case change.

```
5015 \newcommand*{\Glsxtrininlinefullformat}{\Glsxtrfullformat}
```

xtrfullplformat Plural full format without case change.

```
5016 \newcommand*{\glsxtrininlinefullplformat}{\glsxtrfullplformat}
```

inlinefullplformat Plural full format with case change.
5017 \newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}

Redefine \glsentryfull etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the \glsxtrfull set of commands instead.

\glsentryfull
5018 \renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}

\Glsentryfull
5019 \renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}

\glsentryfullpl
5020 \renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}

\Glsentryfullpl
5021 \renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}

gsfirstabbrvfont Font changing command used for the abbreviation on first use or in the full format.
5022 \newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}

gsabbrvdefaultfont Font changing command used for the abbreviation on first use or in the full format.
5023 \newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}

\glsabbrvfont Font changing command used for the abbreviation on subsequent use.
5024 \newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}

gsabbrvdefaultfont
5025 \newcommand*{\glsabbrvdefaultfont}[1]{#1}

\glslongfont Font changing command used for the long form in commands like \glsxtrlong.
5026 \newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}

gslongdefaultfont Default font changing command used for the long form in commands like \glsxtrlong.
5027 \newcommand*{\glslongdefaultfont}[1]{#1}

glsfirstlongfont Font changing command used for the long form on first use or in the full format.
5028 \newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}

gslongdefaultfont
5029 \newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}

gsbrvpluralsuffix Default plural suffix. Allow an alternative default suffix for abbreviations.
5030 \newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}

brvpluralsuffix Default plural suffix.

```
5031 \newcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}
```

\glxtrfull Full form (no case-change).

```
5032 \newrobustcmd*{\glxtrfull}{\@gls@hyp@opt\@ns@glxtrfull}
5033 \newcommand*\ns@glxtrfull[2][\%
5034   \new@ifnextchar[\@glxtr@full{#1}{#2}}{\%
5035     {\@glxtr@full{#1}{#2}[]}}{\%
5036 }
```

\@glxtr@full Low-level macro:

```
5037 \def\@glxtr@full#1#2[#3]{\%
5038   \glsdoifexists{#2}\%
5039   {\%
5040     \glsssetabbrvfmt{\glscategory{#2}}{\%
5041       \let\do@glsc@link@checkfirsthyper\@glsc@link@nocheckfirsthyper
5042       \let\glsc@link@secondoftwo
5043       \let\glsc@link@firstofthree
5044       \let\glsc@link@empty
5045       \def\glscustomtext{\glxtrinlinefullformat{#2}{#3}}{\%

```

What should \glxtrifwasfirstuse be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

```
5046   \glxtrsetupfulldefs
5047   \@glsc@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}\%
5048   }\%
5049   \glspostlinkhook
5050 }
```

trsetupfulldefs

```
5051 \newcommand*{\glxtrsetupfulldefs}{\%
5052   \let\glxtrifwasfirstuse\@firstoftwo
5053 }
```

\Glsxtrfull Full form (first letter uppercase).

```
5054 \newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\@ns@Glsxtrfull}
5055 \newcommand*\ns@Glsxtrfull[2][\%
5056   \new@ifnextchar[\@Glsxtr@full{#1}{#2}}{\%
5057     {\@Glsxtr@full{#1}{#2}[]}}{\%
5058 }
```

\@Glsxtr@full Low-level macro:

```
5059 \def\@Glsxtr@full#1#2[#3]{\%
5060   \glsdoifexists{#2}\%
5061   {\%
5062     \glsssetabbrvfmt{\glscategory{#2}}{\%

```

```

5063 \let\do@gl@s@link@checkfirsthyper\@gl@s@link@nocheckfirsthyper
5064 \let\gl@sifplural\@secondoftwo
5065 \let\gl@scapscase\@secondofthree
5066 \let\gl@sininsert\@empty
5067 \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
5068 \glxtrsetupfulldefs
5069 \@gl@s@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5070 }%
5071 \glspostlinkhook
5072 }

```

\Glsxtrfull Full form (all uppercase).

```

5073 \newrobustcmd*{\Glsxtrfull}{\@gl@s@hyp@opt\@ns@Glsxtrfull}
5074 \newcommand*\ns@Glsxtrfull[2][]{%
5075 \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
5076 {\@Glsxtr@full{#1}{#2}[]}%
5077 }

```

\@Glsxtr@full Low-level macro:

```

5078 \def\@Glsxtr@full#1#2[#3]{%
5079 \gl@doifexists{#2}%
5080 {%
5081 \gl@setabbrvfmt{\gl@category{#2}}%
5082 \let\do@gl@s@link@checkfirsthyper\@gl@s@link@nocheckfirsthyper
5083 \let\gl@sifplural\@secondoftwo
5084 \let\gl@scapscase\@thirdofthree
5085 \let\gl@sininsert\@empty
5086 \def\glscustomtext{\mfirstucMakeUppercase{\glxtrinlinefullformat{#2}{#3}}}%
5087 \glxtrsetupfulldefs
5088 \@gl@s@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5089 }%
5090 \glspostlinkhook
5091 }

```

\glxtrfullpl Plural full form (no case-change).

```

5092 \newrobustcmd*{\glxtrfullpl}{\@gl@s@hyp@opt\@ns@glxtrfullpl}
5093 \newcommand*\ns@glxtrfullpl[2][]{%
5094 \new@ifnextchar[{\@glxtr@fullpl{#1}{#2}}%
5095 {\@glxtr@fullpl{#1}{#2}[]}%
5096 }

```

\@glxtr@fullpl Low-level macro:

```

5097 \def\@glxtr@fullpl#1#2[#3]{%
5098 \gl@doifexists{#2}%
5099 {%
5100 \gl@setabbrvfmt{\gl@category{#2}}%
5101 \let\do@gl@s@link@checkfirsthyper\@gl@s@link@nocheckfirsthyper
5102 \let\gl@sifplural\@firstoftwo
5103 \let\gl@scapscase\@firstofthree

```

```

5104 \let\glsinsert\@empty
5105 \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
5106 \glsxtrsetupfulldefs
5107 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5108 }%
5109 \glspostlinkhook
5110 }

```

`\Glsxtrfullpl` Plural full form (first letter uppercase).

```

5111 \newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\@ns@Glsxtrfullpl}
5112 \newcommand*\ns@Glsxtrfullpl[2][]{%
5113 \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
5114 {\@Glsxtr@fullpl{#1}{#2}[]}%
5115 }

```

`\@Glsxtr@fullpl` Low-level macro:

```

5116 \def\@Glsxtr@fullpl#1#2[#3]{%
5117 \glsdoifexists{#2}%
5118 {%
5119 \glssetabbrvfmt{\glscategory{#2}}%
5120 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5121 \let\glsifplural\@firstoftwo
5122 \let\glscapscase\@secondofthree
5123 \let\glsinsert\@empty
5124 \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
5125 \glsxtrsetupfulldefs
5126 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5127 }%
5128 \glspostlinkhook
5129 }

```

`\GLSxtrfullpl` Plural full form (all upper case).

```

5130 \newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\@ns@GLSxtrfullpl}
5131 \newcommand*\ns@GLSxtrfullpl[2][]{%
5132 \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}%
5133 {\@GLSxtr@fullpl{#1}{#2}[]}%
5134 }

```

`\@GLSxtr@fullpl` Low-level macro:

```

5135 \def\@GLSxtr@fullpl#1#2[#3]{%
5136 \glsdoifexists{#2}%
5137 {%
5138 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5139 \let\glsifplural\@firstoftwo
5140 \let\glscapscase\@thirdofthree
5141 \let\glsinsert\@empty
5142 \def\glscustomtext{%
5143 \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
5144 \glsxtrsetupfulldefs

```

```

5145 \gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5146 }%
5147 \glspostlinkhook
5148 }

```

The short and long forms work in a similar way to acronyms.

`\glsxtrshort`

```

5149 \newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\@ns@glsxtrshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

5150 \newcommand*{\ns@glsxtrshort}[2][\%
5151 \new@ifnextchar[\@glsxtrshort{#1}{#2}]{\@glsxtrshort{#1}{#2}[]}%
5152 }

```

Read in the final optional argument:

```

5153 \def\@glsxtrshort#1#2[#3]{%
5154 \glsdoifexists{#2}%
5155 {%

```

Need to make sure `\glsabbrvfont` is set correctly.

```

5156 \glssetabbrvfmt{\glscategory{#2}}%
5157 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5158 \let\glsxtrifwasfirstuse\@secondoftwo
5159 \let\glsifplural\@secondoftwo
5160 \let\glsapscase\@firstofthree
5161 \let\glsinsert\@empty
5162 \def\glscustomtext{%
5163 \glsabbrvfont{\glsaccessshort{#2}\ifglsxtrininsertinside#3\fi}%
5164 \ifglsxtrininsertinside\else#3\fi
5165 }%
5166 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5167 }%
5168 \glspostlinkhook
5169 }

```

`\Glsxtrshort`

```

5170 \newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\@ns@Glsxtrshort}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

5171 \newcommand*{\ns@Glsxtrshort}[2][\%
5172 \new@ifnextchar[\@Glsxtrshort{#1}{#2}]{\@Glsxtrshort{#1}{#2}[]}%
5173 }

```

Read in the final optional argument:

```

5174 \def\@Glsxtrshort#1#2[#3]{%
5175 \glsdoifexists{#2}%
5176 {%
5177 \glssetabbrvfmt{\glscategory{#2}}%
5178 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5179 \let\glsxtrifwasfirstuse\@secondoftwo

```



```

5180 \let\glsifplural\@secondoftwo
5181 \let\glscapscase\@secondofthree
5182 \let\glsinsert\@empty
5183 \def\glscustomtext{%
5184     \glsabbrvfont{\Glsaccessshort{#2}\ifglxtrinsertinside#3\fi}%
5185     \ifglxtrinsertinside\else#3\fi
5186 }%
5187 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5188 }%
5189 \glspostlinkhook
5190 }

```

\GLSxtrshort

```

5191 \newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\@ns@GLSxtrshort}
    Define the un-starred form. Need to determine if there is a final optional argument
5192 \newcommand*{\ns@GLSxtrshort}[2][{}]{%
5193     \new@ifnextchar[\@GLSxtrshort{#1}{#2}]{\@GLSxtrshort{#1}{#2}[{}]}%
5194 }

    Read in the final optional argument:
5195 \def\@GLSxtrshort#1#2[#3]{%
5196     \glsdoifexists{#2}%
5197     {%
5198         \glssetabbrvfmt{\glscategory{#2}}%
5199         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5200         \let\glxtrifwasfirstuse\@secondoftwo
5201         \let\glsifplural\@secondoftwo
5202         \let\glsapspace\@thirdofthree
5203         \let\glsinsert\@empty
5204         \def\glscustomtext{%
5205             \mfirstucMakeUppercase
5206             {\glsabbrvfont{\Glsaccessshort{#2}\ifglxtrinsertinside#3\fi}%
5207             \ifglxtrinsertinside\else#3\fi
5208         }%
5209     }%
5210     \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5211 }%
5212 \glspostlinkhook
5213 }

```

\glxtrlong

```

5214 \newrobustcmd*{\glxtrlong}{\@gls@hyp@opt\@ns@glxtrlong}
    Define the un-starred form. Need to determine if there is a final optional argument
5215 \newcommand*{\ns@glxtrlong}[2][{}]{%
5216     \new@ifnextchar[\@glxtrlong{#1}{#2}]{\@glxtrlong{#1}{#2}[{}]}%
5217 }

    Read in the final optional argument:
5218 \def\@glxtrlong#1#2[#3]{%

```

```

5219 \glsdoifexists{#2}%
5220 {%
5221   \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5222   \let\glxtrifwasfirstuse\@secondoftwo
5223   \let\gl@ifplural\@secondoftwo
5224   \let\glscapscase\@firstofthree
5225   \let\glinsert\@empty
5226   \def\glscustomtext{%
5227     \glslongfont{\gl@saccesslong{#2}\ifglxtrininsertinside#3\fi}%
5228     \ifglxtrininsertinside\else#3\fi
5229   }%
5230   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5231 }%
5232 \glspostlinkhook
5233 }

```

\Glsxtrlong

```

5234 \newrobustcmd*{\Glsxtrlong}{\@gl@hyp@opt\ns@Glsxtrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

5235 \newcommand*{\ns@Glsxtrlong}[2] [] {%
5236   \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2} []}%
5237 }

```

Read in the final optional argument:

```

5238 \def\@Glsxtrlong#1#2[#3] {%
5239   \glsdoifexists{#2}%
5240   {%
5241     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5242     \let\glxtrifwasfirstuse\@secondoftwo
5243     \let\gl@ifplural\@secondoftwo
5244     \let\glscapscase\@secondofthree
5245     \let\glinsert\@empty
5246     \def\glscustomtext{%
5247       \glslongfont{\Glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
5248       \ifglxtrininsertinside\else#3\fi
5249     }%
5250     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5251   }%
5252   \glspostlinkhook
5253 }

```

\GLSxtrlong

```

5254 \newrobustcmd*{\GLSxtrlong}{\@gl@hyp@opt\ns@GLSxtrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

5255 \newcommand*{\ns@GLSxtrlong}[2] [] {%
5256   \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2} []}%
5257 }

```

Read in the final optional argument:

```

5258 \def\@GLSxtrlong#1#2[#3]{%
5259   \glsdoifexists{#2}%
5260   {%
5261     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5262     \let\glxtrifwasfirstuse\@secondoftwo
5263     \let\glsifplural\@secondoftwo
5264     \let\glscapscase\@thirdofthree
5265     \let\glsinsert\@empty
5266     \def\glscustomtext{%
5267       \mfirstucMakeUppercase
5268       {\glslongfont{\glsaccesslong{#2}\ifglxtrininsertinside#3\fi}%
5269       \ifglxtrininsertinside\else#3\fi
5270     }%
5271   }%
5272   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5273 }%
5274 \glspostlinkhook
5275 }

```

Plural short forms:

\glxtrshortpl

```

5276 \newrobustcmd*{\glxtrshortpl}{\@gl@hyp@opt\@ns@glxtrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

5277 \newcommand*{\ns@glxtrshortpl}[2][ ]{%
5278   \new@ifnextchar[{\@glxtrshortpl{#1}{#2}}{\@glxtrshortpl{#1}{#2}[]}%
5279 }

```

Read in the final optional argument:

```

5280 \def\@glxtrshortpl#1#2[#3]{%
5281   \glsdoifexists{#2}%
5282   {%
5283     \glsetabbrvfmt{\glscategory{#2}}%
5284     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5285     \let\glxtrifwasfirstuse\@secondoftwo
5286     \let\glsifplural\@firstoftwo
5287     \let\glscapscase\@firstofthree
5288     \let\glsinsert\@empty
5289     \def\glscustomtext{%
5290       \glsabbrvfont{\glsaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
5291       \ifglxtrininsertinside\else#3\fi
5292     }%
5293     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5294   }%
5295   \glspostlinkhook
5296 }

```

\Glsxtrshortpl

```

5297 \newrobustcmd*{\Glsxtrshortpl}{\@gl@hyp@opt\@ns@Glsxtrshortpl}

```

Define the un-starred form. Need to determine if there is a final optional argument

```
5298 \newcommand*{\ns@Glsxtrshortpl}[2] [] {%
5299   \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2} []}%
5300 }
```

Read in the final optional argument:

```
5301 \def\@Glsxtrshortpl#1#2[#3] {%
5302   \glsdoifexists{#2}%
5303   {%
5304     \glssetabbrvfmt{\glscategory{#2}}%
5305     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5306     \let\glxtrifwasfirstuse\@secondoftwo
5307     \let\glsifplural\@firstoftwo
5308     \let\glscapscase\@secondofthree
5309     \let\glsinsert\@empty
5310     \def\glscustomtext{%
5311       \glsabbrvfont{\Glsaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
5312       \ifglxtrininsertinside\else#3\fi
5313     }%
5314     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5315   }%
5316   \glspostlinkhook
5317 }
```

\GLSxtrshortpl

```
5318 \newrobustcmd*{\GLSxtrshortpl}{\@gl@hyp@opt\ns@GLSxtrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
5319 \newcommand*{\ns@GLSxtrshortpl}[2] [] {%
5320   \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2} []}%
5321 }
```

Read in the final optional argument:

```
5322 \def\@GLSxtrshortpl#1#2[#3] {%
5323   \glsdoifexists{#2}%
5324   {%
5325     \glssetabbrvfmt{\glscategory{#2}}%
5326     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5327     \let\glxtrifwasfirstuse\@secondoftwo
5328     \let\glsifplural\@firstoftwo
5329     \let\glscapscase\@thirdofthree
5330     \let\glsinsert\@empty
5331     \def\glscustomtext{%
5332       \mfirstucMakeUppercase
5333       {\glsabbrvfont{\Glsaccessshortpl{#2}\ifglxtrininsertinside#3\fi}%
5334       \ifglxtrininsertinside\else#3\fi
5335     }%
5336   }%
5337   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5338 }
```

```

5339 \glspostlinkhook
5340 }

```

Plural long forms:

\glsxtrlongpl

```

5341 \newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\@ns@glxtrlongpl}

Define the un-starred form. Need to determine if there is a final optional argument
5342 \newcommand*{\ns@glxtrlongpl}[2] [] {%
5343   \new@ifnextchar[{\@glxtrlongpl{#1}{#2}}{\@glxtrlongpl{#1}{#2} []}%
5344 }

Read in the final optional argument:
5345 \def\@glxtrlongpl#1#2[#3]{%
5346   \glstoifexists{#2}%
5347   {%
5348     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5349     \let\glxtrifwasfirstuse\@secondoftwo
5350     \let\gl@ifplural\@firstoftwo
5351     \let\glscapscase\@firstofthree
5352     \let\glinsert\@empty
5353     \def\glscustomtext{%
5354       \glslongfont{\glaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
5355       \ifglxtrininsertinside\else#3\fi
5356     }%
5357     \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5358   }%
5359   \glspostlinkhook
5360 }

```

\Glsxtrlongpl

```

5361 \newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\@ns@Glsxtrlongpl}

Define the un-starred form. Need to determine if there is a final optional argument
5362 \newcommand*{\ns@Glsxtrlongpl}[2] [] {%
5363   \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2} []}%
5364 }

Read in the final optional argument:
5365 \def\@Glsxtrlongpl#1#2[#3]{%
5366   \glstoifexists{#2}%
5367   {%
5368     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
5369     \let\glxtrifwasfirstuse\@secondoftwo
5370     \let\gl@ifplural\@firstoftwo
5371     \let\glscapscase\@secondofthree
5372     \let\glinsert\@empty
5373     \def\glscustomtext{%
5374       \glslongfont{\Glsaccesslongpl{#2}\ifglxtrininsertinside#3\fi}%
5375       \ifglxtrininsertinside\else#3\fi

```

```

5376 }%
5377 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5378 }%
5379 \glspostlinkhook
5380 }

```

`\GLSxtrlongpl`

```

5381 \newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\@ns@GLSxtrlongpl}
    Define the un-starred form. Need to determine if there is a final optional argument
5382 \newcommand*{\ns@GLSxtrlongpl}[2] [] {%
5383   \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2} []}%
5384 }

    Read in the final optional argument:
5385 \def\@GLSxtrlongpl#1#2[#3]{%
5386   \glsdoifexists{#2}%
5387   {%
5388     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
5389     \let\glstrifwasfirstuse\@secondoftwo
5390     \let\glsifplural\@firstoftwo
5391     \let\glscapscase\@thirdofthree
5392     \let\glsinsert\@empty
5393     \def\glscustomtext{%
5394       \mfirstucMakeUppercase
5395       {\glsfont{\glsaccesslongpl{#2}\ifglstrinsertinside#3\fi}%
5396       \ifglstrinsertinside\else#3\fi
5397     }%
5398   }%
5399   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
5400 }%
5401 \glspostlinkhook
5402 }

```

`\glssetabbrvfmt` Set the current format for the given category (or the abbreviation category if unset).

```

5403 \newcommand*{\glssetabbrvfmt}[1]{%
5404   \ifcsdef{@glsabbrv@current@#1}%
5405   {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
5406   {\glsxtr@applyabbrvfmt{@glsabbrv@current@abbreviation}}%
5407 }

```

`sxtrgenabbrvfmt` Similar to `\glsgenacfmt`, but for abbreviations.

```

5408 \newcommand*{\glsxtrgenabbrvfmt}{%
5409   \ifdefempty\glscustomtext
5410   {%
5411     \ifglssused\glslabel
5412     {%

```

Subsequent use:

```
5413      \glsifplural
5414      {%
```

Subsequent plural form:

```
5415      \glscapscase
5416      {%
```

Subsequent plural form, don't adjust case:

```
5417      \glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert
5418      }%
5419      {%
```

Subsequent plural form, make first letter upper case:

```
5420      \glsabbrvfont{\Glsaccessshortpl{\glslabel}}\glsinsert
5421      }%
5422      {%
```

Subsequent plural form, all caps:

```
5423      \mfirstucMakeUppercase
5424      {\glsabbrvfont{\glsaccessshortpl{\glslabel}}\glsinsert}%
5425      }%
5426      }%
5427      {%
```

Subsequent singular form

```
5428      \glscapscase
5429      {%
```

Subsequent singular form, don't adjust case:

```
5430      \glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert
5431      }%
5432      {%
```

Subsequent singular form, make first letter upper case:

```
5433      \glsabbrvfont{\Glsaccessshort{\glslabel}}\glsinsert
5434      }%
5435      {%
```

Subsequent singular form, all caps:

```
5436      \mfirstucMakeUppercase
5437      {\glsabbrvfont{\glsaccessshort{\glslabel}}\glsinsert}%
5438      }%
5439      }%
5440      }%
5441      {%
```

First use:

```
5442      \glsifplural
5443      {%
```

First use plural form:

```
5444      \glscapscase
5445      {%
```

First use plural form, don't adjust case:

```
5446 \glstrfullplformat{\glslabel}{\glsinsert}%  
5447 }%  
5448 {%
```

First use plural form, make first letter upper case:

```
5449 \Glsxtrfullplformat{\glslabel}{\glsinsert}%  
5450 }%  
5451 {%
```

First use plural form, all caps:

```
5452 \mfirstucMakeUppercase  
5453 {\glstrfullplformat{\glslabel}{\glsinsert}}%  
5454 }%  
5455 }%  
5456 {%
```

First use singular form

```
5457 \glscapscase  
5458 {%
```

First use singular form, don't adjust case:

```
5459 \glstrfullformat{\glslabel}{\glsinsert}%  
5460 }%  
5461 {%
```

First use singular form, make first letter upper case:

```
5462 \Glsxtrfullformat{\glslabel}{\glsinsert}%  
5463 }%  
5464 {%
```

First use singular form, all caps:

```
5465 \mfirstucMakeUppercase  
5466 {\glstrfullformat{\glslabel}{\glsinsert}}%  
5467 }%  
5468 }%  
5469 }%  
5470 }%  
5471 {%
```

User supplied text.

```
5472 \glscustomtext  
5473 }%  
5474 }
```

1.6.1 Abbreviation Styles Setup

abbreviationstyle

```
5475 \newcommand*{\setabbreviationstyle}[2][abbreviation]{%  
5476 \ifcsundef{@glsabbrv@dispstyle@setup@#2}  
5477 {%
```



```

5478 \PackageError{glossaries-extra}{Undefined abbreviation style ‘#2’}{}%
5479 }%
5480 {%

```

Have abbreviations already been defined for this category?

```

5481 \ifcsstring{@glsabbrv@current@#1}{#2}%
5482 {%

```

Style already set.

```

5483 }%
5484 {%
5485 \def\@glsxtr@dostylewarn{%
5486 \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
5487 {%
5488 \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
5489 style has been switched \MessageBreak
5490 for category ‘#1’, \MessageBreak
5491 but there have already been entries \MessageBreak
5492 defined for this category. Unwanted \MessageBreak
5493 side-effects may result}}%
5494 \@endfortrue
5495 }%
5496 \@glsxtr@dostylewarn

```

Set up the style for the given category.

```

5497 \csdef{@glsabbrv@current@#1}{#2}%
5498 \glsxtr@applyabbrvstyle{#2}%
5499 }%
5500 }%
5501 }

```

applyabbrvstyle Apply the abbreviation style without existence check.

```

5502 \newcommand*{\glsxtr@applyabbrvstyle}[1]{%
5503 \csuse{@glsabbrv@dispstyle@setup@#1}%
5504 \csuse{@glsabbrv@dispstyle@fmts@#1}%
5505 }

```

r@applyabbrvfmt Only apply the style formats.

```

5506 \newcommand*{\glsxtr@applyabbrvfmt}[1]{%
5507 \csuse{@glsabbrv@dispstyle@fmts@#1}%
5508 }

```

abbreviationstyle This is different from `\newacronymstyle`. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```

5509 \newcommand*{\newabbreviationstyle}[3]{%
5510 \ifcsdef{@glsabbrv@dispstyle@setup@#1}
5511 {%
5512 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ already
5513 defined}{}%

```

```

5514 }%
5515 {%
5516 \csdef{@glsabbrv@dispstyle@setup@#1}{%
  Initialise hook to do nothing. The style may change this.
5517 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
5518 #2}%
5519 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
  Assume in-line form is the same as first use. The style may change this.
5520 \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
5521 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
5522 \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
5523 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
5524 #3}%
5525 }%
5526 }

```

breivationstyle

```

5527 \newcommand*{\renewabbreviationstyle}[3]{%
5528 \ifcsundef{@glsabbrv@dispstyle@setup@#1}
5529 {%
5530 \PackageError{glossaries-extra}{Abbreviation style ‘#1’ not defined}{}%
5531 }%
5532 {%
5533 \csdef{@glsabbrv@dispstyle@setup@#1}{%
  Initialise hook to do nothing. The style may change this.
5534 \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
5535 #2}%
5536 \csdef{@glsabbrv@dispstyle@fmts@#1}{%
  Assume in-line form is the same as first use. The style may change this.
5537 \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
5538 \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
5539 \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
5540 \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
5541 #3}%
5542 }%
5543 }

```

breivationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style’s name.

```

5544 \newcommand*{\letabbreviationstyle}[2]{%
5545 \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
5546 \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
5547 }

```

ecated@abbrstyle `\@glsxtr@deprecated@abbrstyle{<old-name>}{<new-name>}`

Define a synonym for a deprecated abbreviation style.

```

5548 \newcommand*{\@GlsXtr@deprecated@abbrstyle}[2]{%
5549   \csdef{@Glsabbrv@dispstyle@setup@#1}{%
5550     \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
5551     \csuse{@Glsabbrv@dispstyle@setup@#2}%
5552   }%
5553   \csletcs{@Glsabbrv@dispstyle@fmts@#1}{@Glsabbrv@dispstyle@fmts@#2}%
5554 }
```

`\GlsXtrWarnDeprecatedAbbrStyle` Generate warning for deprecated style use.

```

5555 \newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
5556   \GlossariesExtraWarning{Deprecated abbreviation style name ‘#1’,
5557     use ‘#2’ instead}%
5558 }
```

`\GlsXtrUseAbbrStyleSetup`

```

5559 \newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
5560   \ifcsundef{@Glsabbrv@dispstyle@setup@#1}%
5561   {%
5562     \PackageError{glossaries-extra}%
5563       {Unknown abbreviation style definitions ‘#1’}{}%
5564   }%
5565   {%
5566     \csname @Glsabbrv@dispstyle@setup@#1\endcsname
5567   }%
5568 }
```

`\GlsXtrUseAbbrStyleFmts`

```

5569 \newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
5570   \ifcsundef{@Glsabbrv@dispstyle@fmts@#1}%
5571   {%
5572     \PackageError{glossaries-extra}%
5573       {Unknown abbreviation style formats ‘#1’}{}%
5574   }%
5575   {%
5576     \csname @Glsabbrv@dispstyle@fmts@#1\endcsname
5577   }%
5578 }
```

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.

```
5579 \newif\ifglxtrinsertinside
5580 \glxtrinsertinsidefalse
```

long-short

```
5581 \newabbreviationstyle{long-short}%
5582 {%
5583   \renewcommand*{\CustomAbbreviationFields}{%
5584     name={\protect\glsabbrvfont{\the\glsshorttok}},
5585     sort={\the\glsshorttok},
5586     first={\protect\glsfirstlongfont{\the\glslongtok}%
5587       \protect\glxtrfullsep{\the\glslabeltok}%
5588       (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
5589     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
5590       \protect\glxtrfullsep{\the\glslabeltok}%
5591       (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%
5592     plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
5593     description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

```
5594 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5595   \glshasattribute{\the\glslabeltok}{regular}%
5596   {%
5597     \glsselattribute{\the\glslabeltok}{regular}{false}%
5598   }%
5599   }%
5600 }%
5601 }%
5602 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
5603 \renewcommand*{\abbrvpluralsuffix}{\glxtrabbrvpluralsuffix}%
5604 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
5605 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
5606 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
5607 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
5608 \renewcommand*{\glxtrfullformat}[2]{%
5609   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
5610   \ifglxtrinsertinside\else##2\fi
5611   \glxtrfullsep{##1}%
5612   (\glsfirstabbrvfont{\glsaccessshort{##1}})%
5613 }%
5614 \renewcommand*{\glxtrfullplformat}[2]{%
5615   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
5616   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5617   (\glsfirstabbrvfont{\glsaccessshortpl{##1}})%
5618 }%
```

```

5619 \renewcommand*{\Glsxtrfullformat}[2]{%
5620   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
5621   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5622   (\glsfirstabbrvfont{\Glsaccessshort{##1}})%
5623 }%
5624 \renewcommand*{\Glsxtrfullplformat}[2]{%
5625   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
5626   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5627   (\glsfirstabbrvfont{\Glsaccessshortpl{##1}})%
5628 }%
5629 }

```

Set this as the default style for general abbreviations:

```
5630 \setabbreviationstyle{long-short}
```

ngshortdescsort

```
5631 \newcommand*{\glxtrlongshortdescsort}{\the\glslongtok\space(\the\glsshorttok)}
```

long-short-desc User supplies description. The long form is included in the name.

```

5632 \newabbreviationstyle{long-short-desc}%
5633 {%
5634   \renewcommand*{\CustomAbbreviationFields}{%
5635     name={\protect\glxtrfullformat{\the\glslabeltok}{}},
5636     sort={\glxtrlongshortdescsort},%
5637     first={\protect\glsfirstlongfont{\the\glslongtok}%
5638       \protect\glxtrfullsep{\the\glslabeltok}%
5639       (\protect\glsfirstabbrvfont{\the\glsshorttok})},%
5640     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
5641       \protect\glxtrfullsep{\the\glslabeltok}%
5642       (\protect\glsfirstabbrvfont{\the\glsshortpltok})},%

```

The text key should only have the short form.

```

5643   text={\protect\glsabbrvfont{\the\glsshorttok}},%
5644   plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
5645 }%

```

Unset the regular attribute if it has been set.

```

5646 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5647   \glshasattribute{\the\glslabeltok}{regular}%
5648   {%
5649     \glissetattribute{\the\glslabeltok}{regular}{false}%
5650   }%
5651   {}%
5652 }%
5653 }%
5654 {%
5655   \GlsXtrUseAbbrStyleFmts{long-short}%
5656 }

```

short-long Short form followed by long form in parenthesis on first use.

```
5657 \newabbreviationstyle{short-long}%
5658 {%
5659   \renewcommand*{\CustomAbbreviationFields}{%
5660     name={\protect\glsabbrvfont{\the\glsshorttok}},
5661     sort={\the\glsshorttok},
5662     description={\the\glslongtok},%
5663     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
5664       \protect\glsxtrfullsep{\the\glslabeltok}%
5665       (\protect\glsfirstlongfont{\the\glslongtok})},%
5666     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
5667       \protect\glsxtrfullsep{\the\glslabeltok}%
5668       (\protect\glsfirstlongfont{\the\glslongpltok})},%
5669     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
5670   }
```

Unset the regular attribute if it has been set.

```
5670   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5671     \glshasattribute{\the\glslabeltok}{regular}%
5672     {%
5673       \glissetattribute{\the\glslabeltok}{regular}{false}%
5674     }%
5675   }%
5676 }%
5677 }%
5678 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
5679 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
5680 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
5681 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
5682 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
5683 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
5684 \renewcommand*{\glsxtrfullformat}[2]{%
5685   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
5686   \ifglsxtrininsertinside\else##2\fi
5687   \glsxtrfullsep{##1}%
5688   (\glsfirstlongfont{\glsaccesslong{##1}})%
5689 }%
5690 \renewcommand*{\glsxtrfullplformat}[2]{%
5691   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
5692   \ifglsxtrininsertinside\else##2\fi
5693   \glsxtrfullsep{##1}%
5694   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
5695 }%
5696 \renewcommand*{\Glsxtrfullformat}[2]{%
5697   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
5698   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5699   (\glsfirstlongfont{\Glsaccesslong{##1}})%
5700 }
```

```

5700 }%
5701 \renewcommand*{\Glsxtrfullplformat}[2]{%
5702   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%
5703   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5704   (\glsfirstlongfont{\Glsaccesslongpl{##1}})%
5705 }%
5706 }

```

`short-long-desc` User supplies description. The long form is included in the name.

```

5707 \newabbreviationstyle{short-long-desc}%
5708 {%
5709   \renewcommand*{\CustomAbbreviationFields}{%
5710     name={\protect\glsxtrfullformat{\the\glslabeltok}{}},
5711     sort={\the\glsshorttok},%
5712     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
5713       \protect\glsxtrfullsep{\the\glslabeltok}%
5714       (\protect\glsfirstlongfont{\the\glslongtok})},%
5715     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
5716       \protect\glsxtrfullsep{\the\glslabeltok}%
5717       (\protect\glsfirstlongfont{\the\glslongpltok})},%
5718     text={\protect\glsabbrvfont{\the\glsshorttok}},%
5719     plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
5720   }%

```

Unset the regular attribute if it has been set.

```

5721 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5722   \glshasattribute{\the\glslabeltok}{regular}%
5723   {%
5724     \glssetattribute{\the\glslabeltok}{regular}{false}%
5725   }%
5726   {}%
5727 }%
5728 }%
5729 {%
5730   \GlsXtrUseAbbrStyleFmts{short-long}%
5731 }

```

`ongfootnotefont` Only used by the “footnote” styles.

```

5732 \newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{##1}}%

```

`ongfootnotefont` Only used by the “footnote” styles.

```

5733 \newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{##1}}%

```

`xtrabbrvfootnote` `\glsxtrabbrvfootnote{<label>}{<long>}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument *<long>* includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```
5734 \newcommand*{\glstrabbrvfootnote}[2]{\footnote{#2}}
```

footnote Short form followed by long form in footnote on first use.

```
5735 \newabbreviationstyle{footnote}%
5736 {%
5737   \renewcommand*{\CustomAbbreviationFields}{%
5738     name={\protect\glsabbrvfont{\the\glsshorttok}},
5739     sort={\the\glsshorttok},
5740     description={\the\glslongtok},%

5741     first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
5742       \protect\glstrabbrvfootnote{\the\glslabeltok}%
5743       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
5744     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
5745       \protect\glstrabbrvfootnote{\the\glslabeltok}%
5746       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
5747     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.

```
5748 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5749   \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
5750   \glshasattribute{\the\glslabeltok}{regular}%
5751   {%
5752     \glssetattribute{\the\glslabeltok}{regular}{false}%
5753   }%
5754   {}%
5755 }%
5756 }%
5757 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```
5758 \renewcommand*{\abbrvpluralsuffix}{\glstrabbrvpluralsuffix}%
5759 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
5760 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
5761 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
5762 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form followed by the long form as a footnote.

```
5763 \renewcommand*{\glstrfullformat}[2]{%
5764   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglstrinsertinside##2\fi}%
5765   \ifglstrinsertinside\else##2\fi
5766   \protect\glstrabbrvfootnote{##1}%
5767   {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
5768 }%
5769 \renewcommand*{\glstrfullplformat}[2]{%

```



```

5770 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5771 \ifglxtrinsertinside\else##2\fi
5772 \protect\glxtrabbrvfootnote{##1}%
5773 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
5774 }%
5775 \renewcommand*{\Glsxtrfullformat}[2]{%
5776 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5777 \ifglxtrinsertinside\else##2\fi
5778 \protect\glxtrabbrvfootnote{##1}%
5779 {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
5780 }%
5781 \renewcommand*{\Glsxtrfullplformat}[2]{%
5782 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5783 \ifglxtrinsertinside\else##2\fi
5784 \protect\glxtrabbrvfootnote{##1}%
5785 {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
5786 }%

```

The first use full form and the inline full form use the short (long) style.

```

5787 \renewcommand*{\glxtrinlinefullformat}[2]{%
5788 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5789 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5790 (\glsfirstlongfootnotefont{\glsaccesslong{##1}})%
5791 }%
5792 \renewcommand*{\glxtrinlinefullplformat}[2]{%
5793 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5794 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5795 (\glsfirstlongfootnotefont{\glsaccesslongpl{##1}})%
5796 }%
5797 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
5798 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5799 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5800 (\glsfirstlongfootnotefont{\glsaccesslong{##1}})%
5801 }%
5802 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
5803 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5804 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5805 (\glsfirstlongfootnotefont{\glsaccesslongpl{##1}})%
5806 }%
5807 }

```

short-footnote

```
5808 \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.

```
5809 \newabbreviationstyle{postfootnote}%
5810 {%

```

```

5811 \renewcommand*{\CustomAbbreviationFields}{%
5812   name={\protect\glsabbrvfont{\the\glsshorttok}},
5813   sort={\the\glsshorttok},
5814   description={\the\glslongtok},%
5815   first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
5816   firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
5817   plural={\protect\glsabbvfont{\the\glsshortpltok}}}%

```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```

5818 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5819   \csdef{glsxtrpostlink\glscategorylabel}{%
5820     \glsxtrifwasfirstuse
5821     {%

```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```

5822       \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}}%
5823       {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}}%
5824   }%
5825   {}%
5826   }%
5827   \glsattribute{\the\glslabeltok}{regular}%
5828   {%
5829   \glssetattribute{\the\glslabeltok}{regular}{false}%
5830   }%
5831   {}%
5832   }%

```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```

5833 \renewcommand*{\glsxtrsetupfulldefs}{%
5834   \let\glsxtrifwasfirstuse\@secondoftwo
5835   }%
5836 }%
5837 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5838 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
5839 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
5840 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
5841 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
5842 \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%

```

The full format displays the short form. The long form is deferred.

```

5843 \renewcommand*{\glsxtrfullformat}[2]{%
5844   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrininsertinside##2\fi}%
5845   \ifglsxtrininsertinside\else##2\fi
5846   }%
5847 \renewcommand*{\glsxtrfullplformat}[2]{%
5848   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrininsertinside##2\fi}%

```

```

5849 \ifglxtrinsertinside\else##2\fi
5850 }%
5851 \renewcommand*{\Glsxtrfullformat}[2]{%
5852 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5853 \ifglxtrinsertinside\else##2\fi
5854 }%
5855 \renewcommand*{\Glsxtrfullplformat}[2]{%
5856 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5857 \ifglxtrinsertinside\else##2\fi
5858 }%

```

The first use full form and the inline full form use the short (long) style.

```

5859 \renewcommand*{\glxtrinlinefullformat}[2]{%
5860 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5861 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5862 (\glsfirstlongfootnotefont{\Glsaccesslong{##1}})%
5863 }%
5864 \renewcommand*{\glxtrinlinefullplformat}[2]{%
5865 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5866 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5867 (\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}})%
5868 }%
5869 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
5870 \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5871 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5872 (\glsfirstlongfootnotefont{\Glsaccesslong{##1}})%
5873 }%
5874 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
5875 \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5876 \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5877 (\glsfirstlongfootnotefont{\Glsaccesslongpl{##1}})%
5878 }%
5879 }

```

rt-postfootnote

```

5880 \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.

```

5881 \newabbreviationstyle{short}%
5882 {%
5883 \renewcommand*{\CustomAbbreviationFields}{%
5884 name={\protect\glsabbrvfont{\the\glsshorttok}},
5885 sort={\the\glsshorttok},
5886 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
5887 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
5888 text={\protect\glsabbrvfont{\the\glsshorttok}},

```

```

5889 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
5890 description={\the\glslongtok}}%
5891 \renewcommand*\GlsXtrPostNewAbbreviation}{%
5892   \glssetattribute{\the\glslabeltok}{regular}{true}}%
5893 }%
5894 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5895 \renewcommand*\abbrvpluralsuffix{\glsxtrabbrvpluralsuffix}%
5896 \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
5897 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvdefaultfont{##1}}%
5898 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongdefaultfont{##1}}%
5899 \renewcommand*\glslongfont[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short form followed by the long form in parentheses.

```

5900 \renewcommand*\glsxtrinlinefullformat[2]{%
5901   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
5902   \ifglsxtrininsertinside##2\fi}%
5903   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5904   (\glsfirstlongfont{\glsaccesslong{##1}})%
5905 }%
5906 \renewcommand*\glsxtrinlinefullplformat[2]{%
5907   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
5908   \ifglsxtrininsertinside##2\fi}%
5909   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5910   (\glsfirstlongfont{\glsaccesslongpl{##1}})%
5911 }%
5912 \renewcommand*\Glsxtrinlinefullformat[2]{%
5913   \protect\glsfirstabbrvfont{\glsaccessshort{##1}}%
5914   \ifglsxtrininsertinside##2\fi}%
5915   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5916   (\glsfirstlongfont{\Glsaccesslong{##1}})%
5917 }%
5918 \renewcommand*\Glsxtrinlinefullplformat[2]{%
5919   \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}%
5920   \ifglsxtrininsertinside##2\fi}%
5921   \ifglsxtrininsertinside\else##2\fi\glsxtrfullsep{##1}%
5922   (\glsfirstlongfont{\Glsaccesslongpl{##1}})%
5923 }%

```

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.

```

5924 \renewcommand*\glsxtrfullformat[2]{%
5925   \glsfirstabbrvfont{\glsaccessshort{##1}}\ifglsxtrininsertinside##2\fi}%
5926   \ifglsxtrininsertinside\else##2\fi
5927 }%
5928 \renewcommand*\glsxtrfullplformat[2]{%
5929   \glsfirstabbrvfont{\glsaccessshortpl{##1}}\ifglsxtrininsertinside##2\fi}%
5930   \ifglsxtrininsertinside\else##2\fi
5931 }%
5932 \renewcommand*\Glsxtrfullformat[2]{%

```

```

5933 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5934 \ifglxtrinsertinside\else##2\fi
5935 }%
5936 \renewcommand*{\Glsxtrfullplformat}[2]{%
5937 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5938 \ifglxtrinsertinside\else##2\fi
5939 }%
5940 }

```

Set this as the default style for acronyms:

```

5941 \setabbreviationstyle[acronym]{short}

```

short-nolong

```

5942 \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.

```

5943 \newabbreviationstyle{short-desc}%
5944 {%
5945 \renewcommand*{\CustomAbbreviationFields}{%
5946 name={\protect\glsxtrinlinefullformat{\the\glslabeltok}{}},
5947 sort={\the\glsshorttok},
5948 first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
5949 firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
5950 text={\protect\glsabbrvfont{\the\glsshorttok}},
5951 plural={\protect\glsabbrvfont{\the\glsshortpltok}},
5952 description={\the\glslongtok}}%
5953 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
5954 \glssetattribute{\the\glslabeltok}{regular}{true}}%
5955 }%
5956 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

5957 \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
5958 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
5959 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
5960 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
5961 \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%

```

The inline full form displays the short format followed by the long form in parentheses.

```

5962 \renewcommand*{\glsxtrinlinefullformat}[2]{%
5963 \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5964 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
5965 (\glsfirstlongfont{\glsaccesslong{##1}})%
5966 }%
5967 \renewcommand*{\glsxtrinlinefullplformat}[2]{%
5968 \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5969 \ifglxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
5970 (\glsfirstlongfont{\glsaccesslongpl{##1}})%
5971 }%

```

```

5972 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
5973   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5974   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5975   (\glsfirstlongfont{\Glsaccesslong{##1}})%
5976 }%
5977 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
5978   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5979   \ifglxtrinsertinside\else##2\fi\glxtrfullsep{##1}%
5980   (\glsfirstlongfont{\Glsaccesslongpl{##1}})%
5981 }%

```

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.

```

5982 \renewcommand*{\glxtrfullformat}[2]{%
5983   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5984   \ifglxtrinsertinside\else##2\fi
5985 }%
5986 \renewcommand*{\glxtrfullplformat}[2]{%
5987   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5988   \ifglxtrinsertinside\else##2\fi
5989 }%
5990 \renewcommand*{\Glsxtrfullformat}[2]{%
5991   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
5992   \ifglxtrinsertinside\else##2\fi
5993 }%
5994 \renewcommand*{\Glsxtrfullplformat}[2]{%
5995   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
5996   \ifglxtrinsertinside\else##2\fi
5997 }%
5998 }

```

ort-nolong-desc

```
5999 \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.

```

6000 \newabbreviationstyle{long-desc}%
6001 {%
6002   \renewcommand*{\CustomAbbreviationFields}{%
6003     name={\protect\protect\glsfirstlongfont{\the\glslongtok}},
6004     sort={\the\glslongtok},
6005     first={\protect\glsfirstlongfont{\the\glslongtok}},
6006     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
6007     text={\the\glslongtok},
6008     plural={\the\glslongpltok}%
6009   }%
6010   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6011     \glssetattribute{\the\glslabeltok}{regular}{true}}%

```

```
6012 }%
6013 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
6014 \renewcommand*\abbrvpluralsuffix{\glstrabbrvpluralsuffix}%
6015 \renewcommand*\glssabbrvfont[1]{\glssabbrvdefaultfont{##1}}%
6016 \renewcommand*\glssfirstabbrvfont[1]{\glssfirstabbrvdefaultfont{##1}}%
6017 \renewcommand*\glssfirstlongfont[1]{\glssfirstlongdefaultfont{##1}}%
6018 \renewcommand*\glsslongfont[1]{\glsslongdefaultfont{##1}}%
```

The inline full form displays the long format followed by the short form in parentheses.

```
6019 \renewcommand*\glssxtrinlinefullformat[2]{%
6020   \glssfirstlongfont{\glssaccesslong{##1}\ifglssxtrininsertinside##2\fi}%
6021   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
6022   (\protect\glssfirstabbrvfont{\glssaccessshort{##1}})%
6023 }%
6024 \renewcommand*\glssxtrinlinefullplformat[2]{%
6025   \glssfirstlongfont{\glssaccesslongpl{##1}\ifglssxtrininsertinside##2\fi}%
6026   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
6027   (\protect\glssfirstabbrvfont{\glssaccessshortpl{##1}})%
6028 }%
6029 \renewcommand*\Glsxtrinlinefullformat[2]{%
6030   \glssfirstlongfont{\Glsaccesslong{##1}\ifglssxtrininsertinside##2\fi}%
6031   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
6032   (\protect\glssfirstabbrvfont{\Glsaccessshort{##1}})%
6033 }%
6034 \renewcommand*\Glsxtrinlinefullplformat[2]{%
6035   \glssfirstlongfont{\Glsaccesslongpl{##1}\ifglssxtrininsertinside##2\fi}%
6036   \ifglssxtrininsertinside\else##2\fi\glssxtrfullsep{##1}%
6037   (\protect\glssfirstabbrvfont{\Glsaccessshortpl{##1}})%
6038 }%
```

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.

```
6039 \renewcommand*\glssxtrfullformat[2]{%
6040   \glssfirstlongfont{\glssaccesslong{##1}\ifglssxtrininsertinside##2\fi}%
6041   \ifglssxtrininsertinside\else##2\fi
6042 }%
6043 \renewcommand*\glssxtrfullplformat[2]{%
6044   \glssfirstlongfont{\glssaccesslongpl{##1}\ifglssxtrininsertinside##2\fi}%
6045   \ifglssxtrininsertinside\else##2\fi
6046 }%
6047 \renewcommand*\Glsxtrfullformat[2]{%
6048   \glssfirstlongfont{\Glsaccesslong{##1}\ifglssxtrininsertinside##2\fi}%
6049   \ifglssxtrininsertinside\else##2\fi
6050 }%
6051 \renewcommand*\Glsxtrfullplformat[2]{%
6052   \glssfirstlongfont{\Glsaccesslongpl{##1}\ifglssxtrininsertinside##2\fi}%
6053   \ifglssxtrininsertinside\else##2\fi
6054 }%
6055 }
```

ng-noshort-desc Provide a synonym that matches similar styles.

```
6056 \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.

```
6057 \newabbreviationstyle{long}%
6058 {%
6059   \renewcommand*{\CustomAbbreviationFields}{%
6060     name={\protect\glsabbrvfont{\the\glsshorttok}},
6061     sort={\the\glsshorttok},
6062     first={\protect\glsfirstlongfont{\the\glslongtok}},
6063     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
6064     text={\the\glslongtok},
6065     plural={\the\glslongpltok},%
6066     description={\the\glslongtok}%
6067   }%
6068   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6069     \glssetattribute{\the\glslabeltok}{regular}{true}}%
6070 }%
6071 {%
6072   \GlsXtrUseAbbrStyleFmts{long-desc}%
6073 }
```

long-noshort Provide a synonym that matches similar styles.

```
6074 \letabbreviationstyle{long-noshort}{long}
```

1.6.3 Predefined Styles (Small Capitals)

These styles use:

\glsxtrscfont

```
6075 \newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

sxtrfirstscfont

```
6076 \newcommand*{\glsxtrfirstscfont}[1]{\glsxtrscfont{#1}}
```

and for the default short form suffix:

\glsxtrscsuffix

```
6077 \newcommand*{\glsxtrscsuffix}{\glstextup{\glsxtrabbrvpluralsuffix}}
```

long-short-sc

```
6078 \newabbreviationstyle{long-short-sc}%
6079 {%
6080   \GlsXtrUseAbbrStyleSetup{long-short}%
6081 }%
6082 {%
```


Mostly as long-short style:

```
6083 \GlsXtrUseAbbrStyleFmts{long-short}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6084 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6085 \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
6086 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
6087 }
```

g-short-sc-desc

```
6088 \newabbreviationstyle{long-short-sc-desc}%
6089 {%
6090 \GlsXtrUseAbbrStyleSetup{long-short-desc}%
6091 }%
6092 {%
```

Mostly as long-short-desc style:

```
6093 \GlsXtrUseAbbrStyleFmts{long-short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6094 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6095 \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
6096 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
6097 }
```

Now the short (long) version

```
6098 \newabbreviationstyle{short-sc-long}%
6099 {%
6100 \GlsXtrUseAbbrStyleSetup{short-long}%
6101 }%
6102 {%
```

Mostly as short-long style:

```
6103 \GlsXtrUseAbbrStyleFmts{short-long}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6104 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6105 \renewcommand*\glsabbrvfont[1]{\glxtrscfont{##1}}%
6106 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstscfont{##1}}%
6107 }
```

As before but user provides description

```
6108 \newabbreviationstyle{short-sc-long-desc}%
6109 {%
6110 \GlsXtrUseAbbrStyleSetup{short-long-desc}%
6111 }%
6112 {%
```

Mostly as short-long-desc style:

```
6113 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6114 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6115 \renewcommand*{\glssabrvfont[1]}{\glxtrscfont{##1}}%
6116 \renewcommand*{\glssfirstabrvfont[1]}{\glxtrfirstscfont{##1}}%
6117 }
```

short-sc

```
6118 \newabbreviationstyle{short-sc}%
6119 {%
6120 \GlsXtrUseAbbrStyleSetup{short-nolong}%
6121 }%
6122 {%
```

Mostly as short style:

```
6123 \GlsXtrUseAbbrStyleFmts{short-nolong}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6124 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6125 \renewcommand*{\glssabrvfont[1]}{\glxtrscfont{##1}}%
6126 \renewcommand*{\glssfirstabrvfont[1]}{\glxtrfirstscfont{##1}}%
6127 }
```

short-sc-nolong

```
6128 \letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
6129 \newabbreviationstyle{short-sc-desc}%
6130 {%
6131 \GlsXtrUseAbbrStyleSetup{short-desc}%
6132 }%
6133 {%
```

Mostly as short style:

```
6134 \GlsXtrUseAbbrStyleFmts{short-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6135 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6136 \renewcommand*{\glssabrvfont[1]}{\glxtrscfont{##1}}%
6137 \renewcommand*{\glssfirstabrvfont[1]}{\glxtrfirstscfont{##1}}%
6138 }
```

-sc-nolong-desc

```
6139 \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.

```
6140 \newabbreviationstyle{long-noshort-sc}%
6141 {%
6142 \GlsXtrUseAbbrStyleSetup{long-noshort}%
6143 }%
6144 {%
```

Mostly as long style:

```
6145 \GlsXtrUseAbbrStyleFmts{long-noshort}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6146 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6147 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
6148 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
6149 }
```

long-sc Backward compatibility:

```
6150 \@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.

```
6151 \newabbreviationstyle{long-noshort-sc-desc}%
6152 {%
6153 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
6154 }%
6155 {%
```

Mostly as long style:

```
6156 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6157 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6158 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
6159 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
6160 }
```

long-desc-sc Backward compatibility:

```
6161 \@glxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
6162 \newabbreviationstyle{short-sc-footnote}%
6163 {%
6164 \GlsXtrUseAbbrStyleSetup{short-footnote}%
6165 }%
6166 {%
```

Mostly as long style:

```
6167 \GlsXtrUseAbbrStyleFmts{short-footnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6168 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrscsuffix}%
6169 \renewcommand*{\glssabrvfont[1]{\glxtrscfont{##1}}}%
6170 \renewcommand*{\glssfirstabrvfont[1]{\glxtrfirstscfont{##1}}}%
6171 }
```

footnote-sc Backward compatibility:

```
6172 \@glxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}
```

sc-postfootnote

```
6173 \newabbreviationstyle{short-sc-postfootnote}%
6174 {%
6175   \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
6176 }%
6177 {%
```

Mostly as long style:

```
6178   \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
6179   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrscsuffix}%
6180   \renewcommand*\glsabbrvfont[1]{\glstrscfont{##1}}%
6181   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstscfont{##1}}%
6182 }
```

postfootnote-sc Backward compatibility:

```
6183 \@glstr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}
```

1.6.4 Predefined Styles (Fake Small Capitals)

These styles require the `relesepackage`, which must be loaded by the user. These styles all use:

`\glstrsmfont`

```
6184 \newcommand*{\glstrsmfont}[1]{\textsmaller{#1}}
```

`glstrfirstsmfont`

```
6185 \newcommand*{\glstrfirstsmfont}[1]{\glstrsmfont{#1}}
```

and for the default short form suffix:

`\glstrsmsuffix`

```
6186 \newcommand*{\glstrsmsuffix}{\glstrabbrvpluralsuffix}
```

long-short-sm

```
6187 \newabbreviationstyle{long-short-sm}%
6188 {%
6189   \GlsXtrUseAbbrStyleSetup{long-short}%
6190 }%
6191 {%
```

Mostly as long-short style:

```
6192   \GlsXtrUseAbbrStyleFmts{long-short}%
6193   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%
6194   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%
6195   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%
6196 }
```

g-short-sm-desc

```
6197 \newabbreviationstyle{long-short-sm-desc}%  
6198 {%  
6199   \GlsXtrUseAbbrStyleSetup{long-short-desc}%  
6200 }%  
6201 {%
```

Mostly as long-short-desc style:

```
6202   \GlsXtrUseAbbrStyleFmts{long-short-desc}%  
6203   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%  
6204   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%  
6205   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%  
6206 }
```

short-sm-long Now the short (long) version

```
6207 \newabbreviationstyle{short-sm-long}%  
6208 {%  
6209   \GlsXtrUseAbbrStyleSetup{short-long}%  
6210 }%  
6211 {%
```

Mostly as short-long style:

```
6212   \GlsXtrUseAbbrStyleFmts{short-long}%  
6213   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%  
6214   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%  
6215   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%  
6216 }
```

rt-sm-long-desc As before but user provides description

```
6217 \newabbreviationstyle{short-sm-long-desc}%  
6218 {%  
6219   \GlsXtrUseAbbrStyleSetup{short-long-desc}%  
6220 }%  
6221 {%
```

Mostly as short-long-desc style:

```
6222   \GlsXtrUseAbbrStyleFmts{short-long-desc}%  
6223   \renewcommand*\glsabbrvfont[1]{\glstrsmfont{##1}}%  
6224   \renewcommand*\glsfirstabbrvfont[1]{\glstrfirstsmfont{##1}}%  
6225   \renewcommand*{\abbrvpluralsuffix}{\protect\glstrsmsuffix}%  
6226 }
```

short-sm

```
6227 \newabbreviationstyle{short-sm}%  
6228 {%  
6229   \GlsXtrUseAbbrStyleSetup{short-nolong}%  
6230 }%  
6231 {%
```

Mostly as short style:

```
6232 \GlsXtrUseAbbrStyleFmts{short-nolong}%
6233 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6234 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6235 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
6236 }
```

short-sm-nolong

```
6237 \letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
6238 \newabbreviationstyle{short-sm-desc}%
6239 {%
6240 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
6241 }%
6242 {%
```

Mostly as short style:

```
6243 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
6244 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6245 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6246 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
6247 }
```

-sm-nolong-desc

```
6248 \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.

```
6249 \newabbreviationstyle{long-noshort-sm}%
6250 {%
6251 \GlsXtrUseAbbrStyleSetup{long-noshort}%
6252 }%
6253 {%
```

Mostly as long style:

```
6254 \GlsXtrUseAbbrStyleFmts{long-noshort}%
6255 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6256 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6257 \renewcommand*{\abbrvpluralsuffix}{\protect\glxtrsmsuffix}%
6258 }
```

long-sm Backward compatibility:

```
6259 \@glxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}
```

noshort-sm-desc The smaller font will only be used if the short form is explicitly invoked through commands like \glsshort.

```
6260 \newabbreviationstyle{long-noshort-sm-desc}%
6261 {%
```

```

6262 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
6263 }%
6264 {%

```

Mostly as long style:

```

6265 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
6266 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6267 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6268 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
6269 }

```

long-desc-sm Backward compatibility:

```

6270 \@glxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}

```

ort-sm-footnote

```

6271 \newabbreviationstyle{short-sm-footnote}%
6272 {%
6273 \GlsXtrUseAbbrStyleSetup{short-footnote}%
6274 }%
6275 {%

```

Mostly as long style:

```

6276 \GlsXtrUseAbbrStyleFmts{short-footnote}%
6277 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6278 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6279 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
6280 }

```

footnote-sm Backward compatibility:

```

6281 \@glxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}

```

sm-postfootnote

```

6282 \newabbreviationstyle{short-sm-postfootnote}%
6283 {%
6284 \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
6285 }%
6286 {%

```

Mostly as long style:

```

6287 \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
6288 \renewcommand*\glsabbrvfont[1]{\glxtrsmfont{##1}}%
6289 \renewcommand*\glsfirstabbrvfont[1]{\glxtrfirstsmfont{##1}}%
6290 \renewcommand*\abbrvpluralsuffix{\protect\glxtrsmsuffix}%
6291 }

```

postfootnote-sm Backward compatibility:

```

6292 \@glxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}

```

1.6.5 Predefined Styles (Emphasized)

These styles use `\emph` for the short form.

`\glsabbrvemfont`

```
6293 \newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%
```

`\glsfirstabbrvemfont`

```
6294 \newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%
```

`\glsfirstlongemfont` Only used by the “long-em” styles.

```
6295 \newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{#1}}%
```

`\glslongemfont` Only used by the “long-em” styles.

```
6296 \newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

`long-short-em`

```
6297 \newabbreviationstyle{long-short-em}%
6298 {%
6299   \GlsXtrUseAbbrStyleSetup{long-short}%
6300 }%
6301 {%
```

Mostly as long-short style:

```
6302   \GlsXtrUseAbbrStyleFmts{long-short}%
6303   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6304 }
```

`long-short-em-desc`

```
6305 \newabbreviationstyle{long-short-em-desc}%
6306 {%
6307   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
6308 }%
6309 {%
```

Mostly as long-short-desc style:

```
6310   \GlsXtrUseAbbrStyleFmts{long-short-desc}%
6311   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6312 }
```

`long-em-short-em`

```
6313 \newabbreviationstyle{long-em-short-em}%
6314 {%
```

`\glslongemfont` is used in the description since `\glsdesc` doesn't set the style.

```
6315   \renewcommand*{\CustomAbbreviationFields}{%
6316     name={\protect\glsabbrvfont{\the\glsshorttok}},
6317     sort={\the\glsshorttok},
6318     first={\protect\glsfirstlongfont{\the\glslongtok}}%
6319     \protect\glsextrfullsep{\the\glslabeltok}}%
```



```

6320    (\protect\glsfirstabbrvfont{\the\glsshorttok}}),%
6321    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}}%
6322    \protect\glstrfullsep{\the\glslabeltok}%
6323    (\protect\glsfirstabbrvfont{\the\glsshortpltok}}),%
6324    plural={\protect\glsabbvfont{\the\glsshortpltok}},%
6325    description={\protect\glslongemfont{\the\glslongtok}}}%

```

Unset the regular attribute if it has been set.

```

6326 \renewcommand*\GlsXtrPostNewAbbreviation{%
6327   \glshasattribute{\the\glslabeltok}{regular}%
6328   {%
6329     \glssetattribute{\the\glslabeltok}{regular}{false}%
6330   }%
6331   {}}%
6332 }%
6333 }%
6334 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6335 \GlsXtrUseAbbrStyleFmts{long-short}%
6336 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6337 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6338 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
6339 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
6340 }

```

m-short-em-desc

```

6341 \newabbreviationstyle{long-em-short-em-desc}%
6342 {%
6343   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
6344 }%
6345 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6346 \GlsXtrUseAbbrStyleFmts{long-short-desc}%
6347 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6348 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6349 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
6350 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
6351 }

```

short-em-long Now the short (long) version

```

6352 \newabbreviationstyle{short-em-long}%
6353 {%
6354   \GlsXtrUseAbbrStyleSetup{short-long}%
6355 }%
6356 {%

```

Mostly as short-long style:

```

6357 \GlsXtrUseAbbrStyleFmts{short-long}%
6358 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%

```

```

6359 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6360 }

```

rt-em-long-desc As before but user provides description

```

6361 \newabbreviationstyle{short-em-long-desc}%
6362 {%
6363   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
6364 }%
6365 {%
  Mostly as short-long-desc style:
6366   \GlsXtrUseAbbrStyleFmts{short-long-desc}%
6367   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6368   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6369   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
6370   \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
6371 }

```

hort-em-long-em

```

6372 \newabbreviationstyle{short-em-long-em}%
6373 {%
  \glslongemfont is used in the description since \glsdesc doesn't set the style.
6374   \renewcommand*{\CustomAbbreviationFields}{%
6375     name={\protect\glsabbrvfont{\the\glsshorttok}},
6376     sort={\the\glsshorttok},
6377     description={\protect\glslongemfont{\the\glslongtok}},%
6378     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
6379     \protect\glsxtrfullsep{\the\glslabeltok}%
6380     (\protect\glsfirstlongfont{\the\glslongtok}}),%
6381     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
6382     \protect\glsxtrfullsep{\the\glslabeltok}%
6383     (\protect\glsfirstlongfont{\the\glslongpltok}}),%
6384     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
  Unset the regular attribute if it has been set.
6385   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6386     \glshasattribute{\the\glslabeltok}{regular}%
6387     {%
6388       \glssetattribute{\the\glslabeltok}{regular}{false}%
6389     }%
6390     {}%
6391   }%
6392 }%
6393 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6394   \GlsXtrUseAbbrStyleFmts{short-long}%
6395   \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6396   \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6397   \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%

```

```

6398 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
6399 }

```

em-long-em-desc

```

6400 \newabbreviationstyle{short-em-long-em-desc}%
6401 {%
6402 \GlsXtrUseAbbrStyleSetup{short-long-desc}%
6403 }%
6404 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6405 \GlsXtrUseAbbrStyleFmts{short-long-desc}%
6406 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6407 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
6408 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
6409 \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
6410 }

```

short-em

```

6411 \newabbreviationstyle{short-em}%
6412 {%
6413 \GlsXtrUseAbbrStyleSetup{short-nolong}%
6414 }%
6415 {%

```

Mostly as short style:

```

6416 \GlsXtrUseAbbrStyleFmts{short-nolong}%
6417 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6418 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
6419 }

```

short-em-nolong

```

6420 \letabbreviationstyle{short-em-nolong}{short-em}

```

short-em-desc

```

6421 \newabbreviationstyle{short-em-desc}%
6422 {%
6423 \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
6424 }%
6425 {%

```

Mostly as short style:

```

6426 \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
6427 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6428 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
6429 }

```

-em-nolong-desc

```

6430 \letabbreviationstyle{short-em-nolong-desc}{short-em-desc}

```

long-noshort-em The short form is explicitly invoked through commands like \glssshort.

```
6431 \newabbreviationstyle{long-noshort-em}%
6432 {%
6433   \GlsXtrUseAbbrStyleSetup{long-noshort}%
6434 }%
6435 {%
```

Mostly as long-noshort style:

```
6436   \GlsXtrUseAbbrStyleFmts{long-noshort}%
6437   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6438   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
6439 }
```

long-em Backward compatibility:

```
6440 \@glsxtr@deprecated@abbrstyle{long-em}{long-noshort-em}
```

g-em-noshort-em The short form is explicitly invoked through commands like \glssshort.

```
6441 \newabbreviationstyle{long-em-noshort-em}%
6442 {%
6443   \renewcommand*{\CustomAbbreviationFields}{%
6444     name={\protect\glsabbrvfont{\the\glssshorttok}},
6445     sort={\the\glssshorttok},
6446     first={\protect\glsfirstlongfont{\the\glslongtok}},
6447     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
6448     text={\the\glslongtok},
6449     plural={\the\glslongpltok},%
6450     description={\protect\glslongemfont{\the\glslongtok}}%
6451   }%
6452   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6453     \glssetattribute{\the\glslabeltok}{regular}{true}}%
6454 }%
6455 {%
```

Mostly as long-noshort style:

```
6456   \GlsXtrUseAbbrStyleFmts{long-noshort}%
6457   \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
6458   \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
6459   \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
6460   \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
6461 }
```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like \glssshort.

```
6462 \newabbreviationstyle{long-noshort-em-desc}%
6463 {%
6464   \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
6465 }%
6466 {%
```

Mostly as long style:

```
6467 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
6468 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6469 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6470 }
```

long-desc-em Backward compatibility:

```
6471 \@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.

```
6472 \newabbreviationstyle{long-em-noshort-em-desc}%
6473 {%
6474 \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
6475 }%
6476 {%
```

Mostly as long style:

```
6477 \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
6478 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6479 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6480 \renewcommand*\glsfirstlongfont[1]{\glsfirstlongemfont{##1}}%
6481 \renewcommand*\glslongfont[1]{\glslongemfont{##1}}%
6482 }
```

ort-em-footnote

```
6483 \newabbreviationstyle{short-em-footnote}%
6484 {%
6485 \GlsXtrUseAbbrStyleSetup{short-footnote}%
6486 }%
6487 {%
```

Mostly as long style:

```
6488 \GlsXtrUseAbbrStyleFmts{short-footnote}%
6489 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6490 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6491 }
```

footnote-em Backward compatibility:

```
6492 \@glsxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}
```

em-postfootnote

```
6493 \newabbreviationstyle{short-em-postfootnote}%
6494 {%
6495 \GlsXtrUseAbbrStyleSetup{short-postfootnote}%
6496 }%
6497 {%
```

Mostly as long style:

```
6498 \GlsXtrUseAbbrStyleFmts{short-postfootnote}%
6499 \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
6500 \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
6501 }
```

postfootnote-em Backward compatibility:

```
6502 \@glsxtr@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:

glsxtruserfield Default is the useri field.

```
6503 \newcommand*\glsxtruserfield{useri}
```

glsxtruserparen The format of the parenthetical information. The first argument is the long/short form. The second argument is the entry's label. If \glscurrentfieldvalue has been defined, then we have at least glossaries v4.23, which makes it easier for the user to adjust this.

```
6504 \ifdef\glscurrentfieldvalue
6505 {
6506   \newcommand*\glsxtruserparen[2]{%
6507     \glsxtrfullsep{#2}%
6508     (#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{})%
6509   }
6510 }
6511 {
6512   \newcommand*\glsxtruserparen[2]{%
6513     \glsxtrfullsep{#2}%
6514     (#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{})%
6515   }
6516 }
```

Font used for short form:

lsabbrvuserfont

```
6517 \newcommand*\glsabbrvuserfont[1]{#1}
```

Font used for short form on first use:

stabbrvuserfont

```
6518 \newcommand*\glsfirstabbrvuserfont[1]{\glsabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
6519 \newcommand*\glslonguserfont[1]{#1}
```

Font used for long form on first use:

rstlonguserfont

```
6520 \newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
6521 \newcommand*{\glsxtrusersuffix}{\glsxtrabbrvpluralsuffix}
```

long-short-user

```
6522 \newabbreviationstyle{long-short-user}%
```

```
6523 {%
```

\glslonguserfont is used in the description since \glsdesc doesn't set the style.

```
6524 \renewcommand*{\CustomAbbreviationFields}{%
```

```
6525   name={\protect\glsabbrvfont{\the\glsshorttok}},
```

```
6526   sort={\the\glsshorttok},
```

```
6527   first={\protect\glsfirstlongfont{\the\glslongtok}}%
```

```
6528   \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}{\the\glslabeltok}},
```

```
6529   firstplural={\protect\glsfirstlongfont{\the\glslongpltok}}%
```

```
6530   \protect\glsxtruserparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}{\the\glslabeltok}},
```

```
6531   plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
```

```
6532   description={\protect\glslonguserfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
6533 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
6534   \glshasattribute{\the\glslabeltok}{regular}%
```

```
6535   {%
```

```
6536     \glissetattribute{\the\glslabeltok}{regular}{false}%
```

```
6537   }%
```

```
6538   {}%
```

```
6539 }%
```

```
6540 }%
```

```
6541 {%
```

In case the user wants to mix and match font styles, these are redefined here.

```
6542 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
```

```
6543 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
```

```
6544 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
```

```
6545 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
```

```
6546 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
6547 \renewcommand*{\glsxtrfullformat}[2]{%
```

```
6548   \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
```

```
6549   \ifglsxtrinsertinside\else##2\fi
```

```
6550   \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
```

```
6551 }%
```

```
6552 \renewcommand*{\glsxtrfullplformat}[2]{%
```

```
6553   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
```

```
6554   \ifglsxtrinsertinside\else##2\fi
```

```
6555   \glsxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
```

```

6556 }%
6557 \renewcommand*{\Glsxtrfullformat}[2]{%
6558   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6559   \ifglxtrinsertinside\else##2\fi
6560   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
6561 }%
6562 \renewcommand*{\Glsxtrfullplformat}[2]{%
6563   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6564   \ifglxtrinsertinside\else##2\fi
6565   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
6566 }%
6567 }

```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```

6568 \newabbreviationstyle{long-postshort-user}%
6569 {%
6570   \renewcommand*{\CustomAbbreviationFields}{%
6571     name={\protect\glsabbrvfont{\the\glsshorttok}},
6572     sort={\the\glsshorttok},
6573     first={\protect\glsfirstlongfont{\the\glslongtok}},%
6574     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%
6575     plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
6576     description={\protect\glslonguserfont{\the\glslongtok}}}%
6577   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6578     \csdef{glxtrpostlink\glscategorylabel}{%
6579       \glxtrifwasfirstuse
6580       {%
6581         \glxtruserparen
6582           {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
6583           {\glslabel}%
6584       }%
6585     }%
6586   }%
6587   \glshasattribute{\the\glslabeltok}{regular}%
6588   {%
6589     \glissetattribute{\the\glslabeltok}{regular}{false}%
6590   }%
6591   {%
6592 }%
6593 }%
6594 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6595 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
6596 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
6597 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
6598 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
6599 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%

```

First use full form:


```

6600 \renewcommand*{\glxtrfullformat}[2]{%
6601   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6602   \ifglxtrinsertinside\else##2\fi
6603 }%
6604 \renewcommand*{\glxtrfullplformat}[2]{%
6605   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6606   \ifglxtrinsertinside\else##2\fi
6607 }%
6608 \renewcommand*{\Glsxtrfullformat}[2]{%
6609   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6610   \ifglxtrinsertinside\else##2\fi
6611 }%
6612 \renewcommand*{\Glsxtrfullplformat}[2]{%
6613   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6614   \ifglxtrinsertinside\else##2\fi
6615 }%

```

In-line format:

```

6616 \renewcommand*{\glxtrinlinefullformat}[2]{%
6617   \glsfirstlongfont{\glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6618   \ifglxtrinsertinside\else##2\fi
6619   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
6620 }%
6621 \renewcommand*{\glxtrinlinefullplformat}[2]{%
6622   \glsfirstlongfont{\glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6623   \ifglxtrinsertinside\else##2\fi
6624   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
6625 }%
6626 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
6627   \glsfirstlongfont{\Glsaccesslong{##1}\ifglxtrinsertinside##2\fi}%
6628   \ifglxtrinsertinside\else##2\fi
6629   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}{##1}%
6630 }%
6631 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
6632   \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglxtrinsertinside##2\fi}%
6633   \ifglxtrinsertinside\else##2\fi
6634   \glxtruserparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}{##1}%
6635 }%
6636 }

```

short-user-desc Like long-postshort-user but the user supplies the description.

```

6637 \newabbreviationstyle{long-postshort-user-desc}%
6638 {%
6639   \renewcommand*{\CustomAbbreviationFields}{%
6640     name={\protect\glsfirstlongfont{\the\glslongtok}%
6641           \protect\glxtruserparen
6642           {\protect\glsabbrvfont{\the\glsshorttok}}{\the\glslabeltok}},
6643     sort={\the\glslongtok},
6644     first={\protect\glsfirstlongfont{\the\glslongtok}},%
6645     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%

```

```

6646 plural={\protect\glsabbvfont{\the\glsshortpltok}}}%
6647 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6648   \csdef{glxtrpostlink\glscategorylabel}{%
6649     \glxtrifwasfirstuse
6650     {%
6651       \glxtruserparen
6652       {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
6653       {\glslabel}%
6654     }%
6655     {}%
6656   }%
6657   \glshasattribute{\the\glslabeltok}{regular}%
6658   {%
6659     \glsssetAttribute{\the\glslabeltok}{regular}{false}%
6660   }%
6661   {}%
6662 }%
6663 }%
6664 {%
6665   \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
6666 }

```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```

6667 \newabbreviationstyle{short-postlong-user}%
6668 {%
6669   \renewcommand*{\CustomAbbreviationFields}{%
6670     name={\protect\glsabbvfont{\the\glsshorttok}},
6671     sort={\the\glsshorttok},
6672     first={\protect\glsfirstlongfont{\the\glslongtok}},%
6673     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%
6674     plural={\protect\glsabbvfont{\the\glsshortpltok}},%
6675     description={\protect\glslonguserfont{\the\glslongtok}}}%
6676   \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6677     \csdef{glxtrpostlink\glscategorylabel}{%
6678       \glxtrifwasfirstuse
6679       {%
6680         \glxtruserparen
6681         {\glsfirstabbrvuserfont{\glsentrylong{\glslabel}}}%
6682         {\glslabel}%
6683       }%
6684       {}%
6685     }%
6686     \glshasattribute{\the\glslabeltok}{regular}%
6687     {%
6688       \glsssetAttribute{\the\glslabeltok}{regular}{false}%
6689     }%
6690     {}%
6691   }%
6692 }%

```

6693 {%

In case the user wants to mix and match font styles, these are redefined here.

```
6694 \renewcommand*{\abbrvpluralsuffix}{\glxtrusersuffix}%
6695 \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
6696 \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
6697 \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
6698 \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

```
6699 \renewcommand*{\glxtrfullformat}[2]{%
6700   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6701   \ifglxtrininsertinside\else##2\fi
6702 }%
6703 \renewcommand*{\glxtrfullplformat}[2]{%
6704   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
6705   \ifglxtrininsertinside\else##2\fi
6706 }%
6707 \renewcommand*{\Glsxtrfullformat}[2]{%
6708   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6709   \ifglxtrininsertinside\else##2\fi
6710 }%
6711 \renewcommand*{\Glsxtrfullplformat}[2]{%
6712   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
6713   \ifglxtrininsertinside\else##2\fi
6714 }%
```

In-line format:

```
6715 \renewcommand*{\glxtrinlinefullformat}[2]{%
6716   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6717   \ifglxtrininsertinside\else##2\fi
6718   \glxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
6719 }%
6720 \renewcommand*{\glxtrinlinefullplformat}[2]{%
6721   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
6722   \ifglxtrininsertinside\else##2\fi
6723   \glxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
6724 }%
6725 \renewcommand*{\Glsxtrinlinefullformat}[2]{%
6726   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrininsertinside##2\fi}%
6727   \ifglxtrininsertinside\else##2\fi
6728   \glxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
6729 }%
6730 \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
6731   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrininsertinside##2\fi}%
6732   \ifglxtrininsertinside\else##2\fi
6733   \glxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
6734 }%
6735 }
```

long-user-desc Like short-postlong-user but leaves the user to specify the description.

```
6736 \newabbreviationstyle{short-postlong-user-desc}%
6737 {%
6738   \renewcommand*{\CustomAbbreviationFields}{%
6739     name={\protect\glsabbrvfont{\the\glsshorttok}%
6740       \protect\glsxtruserparen
6741         {\protect\glsfirstlongfont{\the\glslongpltok}}%
6742         {\the\glslabeltok}},
6743     sort={\the\glsshorttok},
6744     first={\protect\glsfirstlongfont{\the\glslongtok}},%
6745     firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},%
6746     plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
6747 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6748   \csdef{glsxtrpostlink\glscategorylabel}{%
6749     \glsxtrifwasfirstuse
6750     {%
6751       \glsxtruserparen
6752       {\glsfirstabbrvuserfont{\glsentrylong{\glslabel}}}%
6753       {\glslabel}%
6754     }%
6755     {%
6756     }%
6757     \glsattribute{\the\glslabeltok}{regular}%
6758     {%
6759       \glssetattribute{\the\glslabeltok}{regular}{false}%
6760     }%
6761     {}}%
6762   }%
6763 }%
6764 {%
6765   \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
6766 }
```

short-user-desc

```
6767 \newabbreviationstyle{long-short-user-desc}%
6768 {%
6769   \GlsXtrUseAbbrStyleSetup{long-short-desc}%
6770 }%
6771 {%
6772   \GlsXtrUseAbbrStyleFmts{long-short-user}%
6773 }
```

short-long-user

```
6774 \newabbreviationstyle{short-long-user}%
6775 {%
6776   \glslonguserfont is used in the description since \glsdesc doesn't set the style.
6776   \renewcommand*{\CustomAbbreviationFields}{%
6777     name={\protect\glsabbrvfont{\the\glsshorttok}},
```

```

6778     sort={\the\glsshorttok},
6779     description={\protect\glslonguserfont{\the\glslongtok}},%
6780     first={\protect\glsfirstabbrvfont{\the\glsshorttok}}%
6781     \protect\glsxtruserparen{\protect\glslonguserfont{\the\glslongtok}}{\the\glslabeltok}},%
6782     firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}}%
6783     \protect\glsxtruserparen{\protect\glslonguserfont{\the\glslongpltok}}{\the\glslabeltok}},%
6784     plural={\protect\glsabbvfont{\the\glsshortpltok}}}%

```

Unset the regular attribute if it has been set.

```

6785 \renewcommand*{\GlsXtrPostNewAbbreviation}{%
6786   \glshasattribute{\the\glslabeltok}{regular}}%
6787   {%
6788     \glssetattribute{\the\glslabeltok}{regular}{false}}%
6789   }%
6790   {}%
6791 }%
6792 }%
6793 {%

```

In case the user wants to mix and match font styles, these are redefined here.

```

6794 \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
6795 \renewcommand*{\glsabbvfont[1]{\glsabbvuserfont{##1}}}%
6796 \renewcommand*{\glsfirstabbrvfont[1]{\glsfirstabbrvuserfont{##1}}}%
6797 \renewcommand*{\glsfirstlongfont[1]{\glsfirstlonguserfont{##1}}}%
6798 \renewcommand*{\glslongfont[1]{\glslonguserfont{##1}}}%

```

The first use full form and the inline full form are the same for this style.

```

6799 \renewcommand*{\glsxtrfullformat}[2]{%
6800   \glsfirstabbrvfont{\glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
6801   \ifglxtrinsertinside\else##2\fi
6802   \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
6803 }%
6804 \renewcommand*{\glsxtrfullplformat}[2]{%
6805   \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6806   \ifglxtrinsertinside\else##2\fi
6807   \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
6808 }%
6809 \renewcommand*{\Glsxtrfullformat}[2]{%
6810   \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglxtrinsertinside##2\fi}%
6811   \ifglxtrinsertinside\else##2\fi
6812   \glsxtruserparen{\glsfirstlongfont{\glsaccesslong{##1}}}{##1}%
6813 }%
6814 \renewcommand*{\Glsxtrfullplformat}[2]{%
6815   \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglxtrinsertinside##2\fi}%
6816   \ifglxtrinsertinside\else##2\fi
6817   \glsxtruserparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}{##1}%
6818 }%
6819 }

```

-long-user-desc

```

6820 \newabbreviationstyle{short-long-user-desc}%
6821 {%
6822   \GlsXtrUseAbbrStyleSetup{short-long-desc}%
6823 }%
6824 {%
6825   \GlsXtrUseAbbrStyleFmts{short-long-user}%
6826 }

```

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 `\glsentryshort`, instead but this doesn't reflect the formatting since it doesn't include `\glsabbrfont`. 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:

```
6827 \let\@glstr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```

6828 \renewcommand*{\markright}[1]{%
6829   \glstrmarkhook
6830   \@glstr@org@markright{\@glstrinmark#1\@glstrnotinmark}%
6831   \glstrrestoremarkhook
6832 }

```

`\markboth` Save original definition:

```
6833 \let\@glstr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```

6834 \renewcommand*{\markboth}[2]{%
6835   \glxtrmarkhook
6836   \@glxtr@org@markboth
6837   {\@glxtrinmark#1\@glxtrnotinmark}%
6838   {\@glxtrinmark#2\@glxtrnotinmark}%
6839   \glxtrrestoremarkhook
6840 }

```

If this causes a problem provide a simple way of switching back to the original definitions:

sxtrRevertMarks

```

6841 \newcommand*{\glxtrRevertMarks}{%
6842   \let\markright\@glxtr@org@markright
6843   \let\markboth\@glxtr@org@markboth
6844 }

```

\glxtrifinmark

```

6845 \newcommand*{\glxtrifinmark}[2]{#2}

```

\@glxtrinmark

```

6846 \newrobustcmd*{\@glxtrinmark}{%
6847   \let\glxtrifinmark\@firstoftwo
6848 }

```

glxtrnotinmark

```

6849 \newrobustcmd*{\@glxtrnotinmark}{%
6850   \let\glxtrifinmark\@secondoftwo
6851 }

```

\glxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```

6852 \newcommand*{\glxtrmarkhook}{%

```

Save current definitions:

```

6853   \let\@glxtr@org@MakeUppercase\MakeUppercase
6854   \let\@glxtr@org@glxtrtitleshort\glxtrtitleshort
6855   \let\@glxtr@org@glxtrtitleshortpl\glxtrtitleshortpl
6856   \let\@glxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
6857   \let\@glxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
6858   \let\@glxtr@org@glxtrtitletext\glxtrtitletext
6859   \let\@glxtr@org@Glsxtrtitletext\Glsxtrtitletext
6860   \let\@glxtr@org@glxtrtitleplural\glxtrtitleplural
6861   \let\@glxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
6862   \let\@glxtr@org@glxtrtitlefirst\glxtrtitlefirst
6863   \let\@glxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
6864   \let\@glxtr@org@glxtrtitlefirstplural\glxtrtitlefirstplural
6865   \let\@glxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
6866   \let\@glxtr@org@glxtrtitlelong\glxtrtitlelong
6867   \let\@glxtr@org@glxtrtitlelongpl\glxtrtitlelongpl

```

```

6868 \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
6869 \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
6870 \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
6871 \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
6872 \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
6873 \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl

```

New definitions

```

6874 \let\glsxtrifinmark\@firstoftwo
6875 \let\MakeUppercase\MakeTextUppercase
6876 \let\glsxtrtitleshort\glsxtrheadshort
6877 \let\glsxtrtitleshortpl\glsxtrheadshortpl
6878 \let\Glsxtrtitleshort\Glsxtrheadshort
6879 \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
6880 \let\glsxtrtitletext\glsxtrheadtext
6881 \let\Glsxtrtitletext\Glsxtrheadtext
6882 \let\glsxtrtitleplural\glsxtrheadplural
6883 \let\Glsxtrtitleplural\Glsxtrheadplural
6884 \let\glsxtrtitlefirst\glsxtrheadfirst
6885 \let\Glsxtrtitlefirst\Glsxtrheadfirst
6886 \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
6887 \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
6888 \let\glsxtrtitlelong\glsxtrheadlong
6889 \let\glsxtrtitlelongpl\glsxtrheadlongpl
6890 \let\Glsxtrtitlelong\Glsxtrheadlong
6891 \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
6892 \let\glsxtrtitlefull\glsxtrheadfull
6893 \let\glsxtrtitlefullpl\glsxtrheadfullpl
6894 \let\Glsxtrtitlefull\Glsxtrheadfull
6895 \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
6896 }

```

`\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.)

```

6897 \newcommand*{\glsxtrrestoremarkhook}{%
6898 \let\glsxtrifinmark\@secondoftwo
6899 \let\MakeUppercase\@glsxtr@org@MakeUppercase
6900 \let\glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
6901 \let\glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
6902 \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
6903 \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
6904 \let\glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
6905 \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
6906 \let\glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
6907 \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
6908 \let\glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
6909 \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst

```



```

6910 \let\glxstrtitlefirstplural\@glxstr@org@glxstrtitlefirstplural
6911 \let\Glsxtrtitlefirstplural\@glxstr@org@Glsxtrtitlefirstplural
6912 \let\glxstrtitlelong\@glxstr@org@glxstrtitlelong
6913 \let\glxstrtitlelongpl\@glxstr@org@glxstrtitlelongpl
6914 \let\Glsxtrtitlelong\@glxstr@org@Glsxtrtitlelong
6915 \let\Glsxtrtitlelongpl\@glxstr@org@Glsxtrtitlelongpl
6916 \let\glxstrtitlefull\@glxstr@org@glxstrtitlefull
6917 \let\glxstrtitlefullpl\@glxstr@org@glxstrtitlefullpl
6918 \let\Glsxtrtitlefull\@glxstr@org@Glsxtrtitlefull
6919 \let\Glsxtrtitlefullpl\@glxstr@org@Glsxtrtitlefullpl
6920 }

```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

glxstrheadshort Command used to display short form in the page header.

```

6921 \newcommand*{\glxstrheadshort}[1]{%
6922 \protect\NoCaseChange
6923 {%
6924 \gl@ifattribute{#1}{headuc}{true}%
6925 {%
6926 \GLSxtrshort[noindex,hyper=false]{#1}[]%
6927 }%
6928 {%
6929 \glxstrshort[noindex,hyper=false]{#1}[]%
6930 }%
6931 }%
6932 }

```

lsxstrtitleshort Command to display short form of abbreviation in section title and table of contents.

```

6933 \newrobustcmd*{\lsxstrtitleshort}[1]{%
6934 \glxstrshort[noindex,hyper=false]{#1}[]%
6935 }

```

sxstrheadshortpl 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.

```

6936 \newcommand*{\glxstrheadshortpl}[1]{%
6937 \protect\NoCaseChange
6938 {%
6939 \gl@ifattribute{#1}{headuc}{true}%
6940 {%
6941 \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
6942 }%
6943 {%
6944 \glxstrshortpl[noindex,hyper=false]{#1}[]%
6945 }%
6946 }%
6947 }

```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents.

```

6948 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
6949   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
6950 }

```

`Glsxtrheadshort` Command used to display short form in the page header with the first letter converted to upper case.

```

6951 \newcommand*{\Glsxtrheadshort}[1]{%
6952   \protect\NoCaseChange
6953   {%
6954     \glsifattribute{#1}{headuc}{true}%
6955     {%
6956       \Glsxtrshort[noindex,hyper=false]{#1}[]%
6957     }%
6958     {%
6959       \Glsxtrshort[noindex,hyper=false]{#1}[]%
6960     }%
6961   }%
6962 }

```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

6963 \newrobustcmd*{\Glsxtrtitleshort}[1]{%
6964   \Glsxtrshort[noindex,hyper=false]{#1}[]%
6965 }

```

`sxtrheadshortpl` Command used to display plural short form in the page header with the first letter converted to upper case.

```

6966 \newcommand*{\Glsxtrheadshortpl}[1]{%
6967   \protect\NoCaseChange
6968   {%
6969     \glsifattribute{#1}{headuc}{true}%
6970     {%
6971       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
6972     }%
6973     {%
6974       \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
6975     }%
6976   }%
6977 }

```

`xtrtitleshortpl` Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

6978 \newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
6979   \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
6980 }

```

`\glsxtrheadtext` As above but for the text value.

```

6981 \newcommand*{\glxtrheadtext}[1]{%
6982   \protect\NoCaseChange
6983   {%
6984     \glsifattribute{#1}{headuc}{true}%
6985     {%
6986       \GLStext[noindex,hyper=false]{#1}[]%
6987     }%
6988     {%
6989       \glstext[noindex,hyper=false]{#1}[]%
6990     }%
6991   }%
6992 }

```

glxtrtitletext Command to display text value in section title and table of contents.

```

6993 \newrobustcmd*{\glxtrtitletext}[1]{%
6994   \glstext[noindex,hyper=false]{#1}[]%
6995 }

```

\Glsxtrheadtext First letter converted to upper case

```

6996 \newcommand*{\Glsxtrheadtext}[1]{%
6997   \protect\NoCaseChange
6998   {%
6999     \glsifattribute{#1}{headuc}{true}%
7000     {%
7001       \GLStext[noindex,hyper=false]{#1}[]%
7002     }%
7003     {%
7004       \Glstext[noindex,hyper=false]{#1}[]%
7005     }%
7006   }%
7007 }

```

Glsxtrtitletext Command to display text value in section title and table of contents with the first letter changed to upper case.

```

7008 \newrobustcmd*{\Glsxtrtitletext}[1]{%
7009   \Glstext[noindex,hyper=false]{#1}[]%
7010 }

```

lsxtrheadplural As above but for the plural value.

```

7011 \newcommand*{\lsxtrheadplural}[1]{%
7012   \protect\NoCaseChange
7013   {%
7014     \glsifattribute{#1}{headuc}{true}%
7015     {%
7016       \GLSplural[noindex,hyper=false]{#1}[]%
7017     }%
7018     {%
7019       \glsplural[noindex,hyper=false]{#1}[]%
7020     }%

```

```

7021 }%
7022 }

```

sxtrtitleplural Command to display plural value in section title and table of contents.

```

7023 \newrobustcmd*{\glsxtrtitleplural}[1]{%
7024   \glsplural[noindex,hyper=false]{#1}[]%
7025 }

```

lsxtrheadplural Convert first letter to upper case.

```

7026 \newcommand*{\Glsxtrheadplural}[1]{%
7027   \protect\NoCaseChange
7028   {%
7029     \gl@ifattribute{#1}{headuc}{true}%
7030     {%
7031       \GLSplural[noindex,hyper=false]{#1}[]%
7032     }%
7033     {%
7034       \Glsplural[noindex,hyper=false]{#1}[]%
7035     }%
7036   }%
7037 }

```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```

7038 \newrobustcmd*{\Glsxtrtitleplural}[1]{%
7039   \Glsplural[noindex,hyper=false]{#1}[]%
7040 }

```

glxtrheadfirst As above but for the first value.

```

7041 \newcommand*{\glxtrheadfirst}[1]{%
7042   \protect\NoCaseChange
7043   {%
7044     \gl@ifattribute{#1}{headuc}{true}%
7045     {%
7046       \GLSfirst[noindex,hyper=false]{#1}[]%
7047     }%
7048     {%
7049       \glsfirst[noindex,hyper=false]{#1}[]%
7050     }%
7051   }%
7052 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```

7053 \newrobustcmd*{\glsxtrtitlefirst}[1]{%
7054   \glsfirst[noindex,hyper=false]{#1}[]%
7055 }

```

Glsxtrheadfirst First letter converted to upper case

```

7056 \newcommand*{\Glsxtrheadfirst}[1]{%
7057   \protect\NoCaseChange
7058   {%
7059     \glsifattribute{#1}{headuc}{true}%
7060     {%
7061       \GLSfirst[noindex,hyper=false]{#1}[]%
7062     }%
7063     {%
7064       \Glsfirst[noindex,hyper=false]{#1}[]%
7065     }%
7066   }%
7067 }

```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```

7068 \newrobustcmd*{\Glsxtrtitlefirst}[1]{%
7069   \GLSfirst[noindex,hyper=false]{#1}[]%
7070 }

```

headfirstplural As above but for the firstplural value.

```

7071 \newcommand*{\Glsxtrheadfirstplural}[1]{%
7072   \protect\NoCaseChange
7073   {%
7074     \glsifattribute{#1}{headuc}{true}%
7075     {%
7076       \GLSfirstplural[noindex,hyper=false]{#1}[]%
7077     }%
7078     {%
7079       \glsfirstplural[noindex,hyper=false]{#1}[]%
7080     }%
7081   }%
7082 }

```

itlefirstplural Command to display firstplural value in section title and table of contents.

```

7083 \newrobustcmd*{\Glsxtritlefirstplural}[1]{%
7084   \glsfirstplural[noindex,hyper=false]{#1}[]%
7085 }

```

headfirstplural First letter converted to upper case

```

7086 \newcommand*{\Glsxtrheadfirstplural}[1]{%
7087   \protect\NoCaseChange
7088   {%
7089     \glsifattribute{#1}{headuc}{true}%
7090     {%
7091       \GLSfirstplural[noindex,hyper=false]{#1}[]%
7092     }%
7093     {%
7094       \Glsfirstplural[noindex,hyper=false]{#1}[]%
7095     }%

```

```

7096 }%
7097 }

```

`\titlefirstplural` Command to display first value in section title and table of contents with the first letter changed to upper case.

```

7098 \newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
7099   \Glsfirstplural[noindex,hyper=false]{#1}[]%
7100 }

```

`\glxstrheadlong` Command used to display long form in the page header.

```

7101 \newcommand*{\glxstrheadlong}[1]{%
7102   \protect\NoCaseChange
7103   {%
7104     \glsifattribute{#1}{headuc}{true}%
7105     {%
7106       \GLSxtrlong[noindex,hyper=false]{#1}[]%
7107     }%
7108     {%
7109       \glxstrlong[noindex,hyper=false]{#1}[]%
7110     }%
7111   }%
7112 }

```

`\glxstrtitlelong` Command to display long form of abbreviation in section title and table of contents.

```

7113 \newrobustcmd*{\glxstrtitlelong}[1]{%
7114   \glxstrlong[noindex,hyper=false]{#1}[]%
7115 }

```

`\glxstrheadlongpl` 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.

```

7116 \newcommand*{\glxstrheadlongpl}[1]{%
7117   \protect\NoCaseChange
7118   {%
7119     \glsifattribute{#1}{headuc}{true}%
7120     {%
7121       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
7122     }%
7123     {%
7124       \glxstrlongpl[noindex,hyper=false]{#1}[]%
7125     }%
7126   }%
7127 }

```

`\glxstrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents.

```

7128 \newrobustcmd*{\glxstrtitlelongpl}[1]{%
7129   \glxstrlongpl[noindex,hyper=false]{#1}[]%
7130 }

```

`\Glsxtrheadlong` Command used to display long form in the page header with the first letter converted to upper case.

```

7131 \newcommand*{\Glsxtrheadlong}[1]{%
7132   \protect\NoCaseChange
7133   {%
7134     \glsifattribute{#1}{headuc}{true}%
7135     {%
7136       \GLSxtrlong[noindex,hyper=false]{#1}[]%
7137     }%
7138     {%
7139       \Glsxtrlong[noindex,hyper=false]{#1}[]%
7140     }%
7141   }%
7142 }
```

`Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

7143 \newrobustcmd*{\Glsxtrtitlelong}[1]{%
7144   \Glsxtrlong[noindex,hyper=false]{#1}[]%
7145 }
```

`lslxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```

7146 \newcommand*{\Glsxtrheadlongpl}[1]{%
7147   \protect\NoCaseChange
7148   {%
7149     \glsifattribute{#1}{headuc}{true}%
7150     {%
7151       \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
7152     }%
7153     {%
7154       \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
7155     }%
7156   }%
7157 }
```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```

7158 \newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
7159   \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
7160 }
```

`\glsxtrheadfull` Command used to display full form in the page header.

```

7161 \newcommand*{\glsxtrheadfull}[1]{%
7162   \protect\NoCaseChange
7163   {%
7164     \glsifattribute{#1}{headuc}{true}%
7165     {%
```

```

7166     \GLSxtrfull[noindex,hyper=false]{#1}[]%
7167 }%
7168 {%
7169     \glsxtrfull[noindex,hyper=false]{#1}[]%
7170 }%
7171 }%
7172 }

```

glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents.

```

7173 \newrobustcmd*{\glsxtrtitlefull}[1]{%
7174   \glsxtrfull[noindex,hyper=false]{#1}[]%
7175 }

```

glsxtrheadfullpl 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.

```

7176 \newcommand*{\glsxtrheadfullpl}[1]{%
7177   \protect\NoCaseChange
7178   {%
7179     \glsifattribute{#1}{headuc}{true}%
7180     {%
7181       \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
7182     }%
7183     {%
7184       \glsxtrfullpl[noindex,hyper=false]{#1}[]%
7185     }%
7186   }%
7187 }

```

glsxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```

7188 \newrobustcmd*{\glsxtrtitlefullpl}[1]{%
7189   \glsxtrfullpl[noindex,hyper=false]{#1}[]%
7190 }

```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```

7191 \newcommand*{\Glsxtrheadfull}[1]{%
7192   \protect\NoCaseChange
7193   {%
7194     \glsifattribute{#1}{headuc}{true}%
7195     {%
7196       \GLSxtrfull[noindex,hyper=false]{#1}[]%
7197     }%
7198     {%
7199       \Glsxtrfull[noindex,hyper=false]{#1}[]%
7200     }%
7201   }%
7202 }

```


`\Glsxtrtitlefull` Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
7203 \newrobustcmd*{\Glsxtrtitlefull}[1]{%
7204   \Glsxtrfull[noindex,hyper=false]{#1}[]%
7205 }
```

`\Glsxtrheadfullpl` Command used to display plural full form in the page header with the first letter converted to upper case.

```
7206 \newcommand*{\Glsxtrheadfullpl}[1]{%
7207   \protect\NoCaseChange
7208   {%
7209     \glsifattribute{#1}{headuc}{true}%
7210     {%
7211       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
7212     }%
7213     {%
7214       \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
7215     }%
7216   }%
7217 }
```

`\Glsxtrtitlefullpl` Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
7218 \newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
7219   \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
7220 }
```

`\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.

```
7221 \ifdef\texorpdfstring
7222 {
7223   \newcommand*{\glsfmtshort}[1]{%
7224     \texorpdfstring
7225       {\glsxtrtitleshort{#1}}%
7226       {\glsentryshort{#1}}%
7227   }
7228 }
7229 {
7230   \newcommand*{\glsfmtshort}[1]{%
7231     \glsxtrtitleshort{#1}}
7232 }
```

Similarly for the plural version.

`\glsfmtshortpl`

```
7233 \ifdef\texorpdfstring
7234 {
7235   \newcommand*{\glsfmtshortpl}[1]{%
```

```

7236     \texorpdfstring
7237     {\glxtrtitleshortpl{#1}}}%
7238     {\glsentryshortpl{#1}}}%
7239 }
7240 }
7241 {
7242   \newcommand*{\glsfmtshortpl}[1]{%
7243     \glxtrtitleshortpl{#1}}
7244 }

```

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).

```

7245 \ifdef\texorpdfstring
7246 {
7247   \newcommand*{\Glsfmtshort}[1]{%
7248     \texorpdfstring
7249     {\Glsxtrtitleshort{#1}}}%
7250     {\glsentryshort{#1}}}%
7251 }
7252 }
7253 {
7254   \newcommand*{\Glsfmtshort}[1]{%
7255     \Glsxtrtitleshort{#1}}
7256 }

```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```

7257 \ifdef\texorpdfstring
7258 {
7259   \newcommand*{\Glsfmtshortpl}[1]{%
7260     \texorpdfstring
7261     {\Glsxtrtitleshortpl{#1}}}%
7262     {\glsentryshortpl{#1}}}%
7263 }
7264 }
7265 {
7266   \newcommand*{\Glsfmtshortpl}[1]{%
7267     \Glsxtrtitleshortpl{#1}}
7268 }

```

`\glsfmttext` As above but for the text value.

```

7269 \ifdef\texorpdfstring
7270 {
7271   \newcommand*{\glsfmttext}[1]{%
7272     \texorpdfstring
7273     {\glxtrtitletext{#1}}}%
7274     {\glsentrytext{#1}}}%
7275 }

```

```

7276 }
7277 {
7278   \newcommand*{\glsfmttext}[1]{%
7279     \glsxtrtitletext{#1}}
7280 }

```

`\Glsfmttext` First letter converted to upper case.

```

7281 \ifdef\texorpdfstring
7282 {
7283   \newcommand*{\Glsfmttext}[1]{%
7284     \texorpdfstring
7285       {\Glsxtrtitletext{#1}}%
7286       {\glsentrytext{#1}}%
7287   }
7288 }
7289 {
7290   \newcommand*{\Glsfmttext}[1]{%
7291     \Glsxtrtitletext{#1}}
7292 }

```

`\glsfmtplural` As above but for the plural value.

```

7293 \ifdef\texorpdfstring
7294 {
7295   \newcommand*{\glsfmtplural}[1]{%
7296     \texorpdfstring
7297       {\glsxtrtitleplural{#1}}%
7298       {\glsentryplural{#1}}%
7299   }
7300 }
7301 {
7302   \newcommand*{\glsfmtplural}[1]{%
7303     \glsxtrtitleplural{#1}}
7304 }

```

`\Glsfmtplural` First letter converted to upper case.

```

7305 \ifdef\texorpdfstring
7306 {
7307   \newcommand*{\Glsfmtplural}[1]{%
7308     \texorpdfstring
7309       {\Glsxtrtitleplural{#1}}%
7310       {\glsentryplural{#1}}%
7311   }
7312 }
7313 {
7314   \newcommand*{\Glsfmtplural}[1]{%
7315     \Glsxtrtitleplural{#1}}
7316 }

```

`\glsfmtfirst` As above but for the first value.

```

7317 \ifdef\texorpdfstring
7318 {
7319   \newcommand*\glsfmtfirst}[1]{%
7320     \texorpdfstring
7321     {\glsxtrtitlefirst{#1}}%
7322     {\glsentryfirst{#1}}%
7323   }
7324 }
7325 {
7326   \newcommand*\glsfmtfirst}[1]{%
7327     \glsxtrtitlefirst{#1}}
7328 }

```

`\Glsfmtfirst` First letter converted to upper case.

```

7329 \ifdef\texorpdfstring
7330 {
7331   \newcommand*\Glsfmtfirst}[1]{%
7332     \texorpdfstring
7333     {\Glsxtrtitlefirst{#1}}%
7334     {\glsentryfirst{#1}}%
7335   }
7336 }
7337 {
7338   \newcommand*\Glsfmtfirst}[1]{%
7339     \Glsxtrtitlefirst{#1}}
7340 }

```

`\glsfmtfirstpl` As above but for the firstplural value.

```

7341 \ifdef\texorpdfstring
7342 {
7343   \newcommand*\glsfmtfirstpl}[1]{%
7344     \texorpdfstring
7345     {\glsxtrtitlefirstplural{#1}}%
7346     {\glsentryfirstplural{#1}}%
7347   }
7348 }
7349 {
7350   \newcommand*\glsfmtfirstpl}[1]{%
7351     \glsxtrtitlefirstplural{#1}}
7352 }

```

`\Glsfmtfirstpl` First letter converted to upper case.

```

7353 \ifdef\texorpdfstring
7354 {
7355   \newcommand*\Glsfmtfirstpl}[1]{%
7356     \texorpdfstring
7357     {\Glsxtrtitlefirstplural{#1}}%
7358     {\glsentryfirstplural{#1}}%
7359   }

```

```

7360 }
7361 {
7362   \newcommand*{\Glsfmtfirstpl}[1]{%
7363     \Glsxtrtitlefirstplural{#1}}
7364 }

```

`\glsfmtlong` As above but for the long value.

```

7365 \ifdef\texorpdfstring
7366 {
7367   \newcommand*{\glsfmtlong}[1]{%
7368     \texorpdfstring
7369     {\Glsxtrtitlelong{#1}}%
7370     {\glsentrylong{#1}}%
7371   }
7372 }
7373 {
7374   \newcommand*{\glsfmtlong}[1]{%
7375     \Glsxtrtitlelong{#1}}
7376 }

```

`\Glsfmtlong` First letter converted to upper case.

```

7377 \ifdef\texorpdfstring
7378 {
7379   \newcommand*{\Glsfmtlong}[1]{%
7380     \texorpdfstring
7381     {\Glsxtrtitlelong{#1}}%
7382     {\glsentrylong{#1}}%
7383   }
7384 }
7385 {
7386   \newcommand*{\Glsfmtlong}[1]{%
7387     \Glsxtrtitlelong{#1}}
7388 }

```

`\glsfmtlongpl` As above but for the longplural value.

```

7389 \ifdef\texorpdfstring
7390 {
7391   \newcommand*{\glsfmtlongpl}[1]{%
7392     \texorpdfstring
7393     {\Glsxtrtitlelongpl{#1}}%
7394     {\glsentrylongpl{#1}}%
7395   }
7396 }
7397 {
7398   \newcommand*{\glsfmtlongpl}[1]{%
7399     \Glsxtrtitlelongpl{#1}}
7400 }

```

`\Glsfmtlongpl` First letter converted to upper case.

```

7401 \ifdef\texorpdfstring
7402 {
7403   \newcommand*\Glsfmtlongpl[1]{%
7404     \texorpdfstring
7405     {\Glsxtrtitlelongpl{#1}}%
7406     {\glsentrylongpl{#1}}%
7407   }
7408 }
7409 {
7410   \newcommand*\Glsfmtlongpl[1]{%
7411     \Glsxtrtitlelongpl{#1}}
7412 }

```

`\glsfmtfull` In-line full format.

```

7413 \ifdef\texorpdfstring
7414 {
7415   \newcommand*\glsfmtfull[1]{%
7416     \texorpdfstring
7417     {\glsxtrtitlefull{#1}}%
7418     {\glsxtrinlinefullformat{#1}{}}%
7419   }
7420 }
7421 {
7422   \newcommand*\glsfmtfull[1]{%
7423     \glsxtrtitlefull{#1}}
7424 }

```

`\Glsfmtfull` First letter converted to upper case.

```

7425 \ifdef\texorpdfstring
7426 {
7427   \newcommand*\Glsfmtfull[1]{%
7428     \texorpdfstring
7429     {\Glsxtrtitlefull{#1}}%
7430     {\Glsxtrinlinefullformat{#1}{}}%
7431   }
7432 }
7433 {
7434   \newcommand*\Glsfmtfull[1]{%
7435     \Glsxtrtitlefull{#1}}
7436 }

```

`\glsfmtfullpl` In-line full plural format.

```

7437 \ifdef\texorpdfstring
7438 {
7439   \newcommand*\glsfmtfullpl[1]{%
7440     \texorpdfstring
7441     {\glsxtrtitlefullpl{#1}}%
7442     {\glsxtrinlinefullplformat{#1}{}}%
7443   }

```

```

7444 }
7445 {
7446   \newcommand*{\Glsfmtfullpl}[1]{%
7447     \Glsxtrtitlefullpl{#1}}
7448 }

```

`\Glsfmtfullpl` First letter converted to upper case.

```

7449 \ifdef\texorpdfstring
7450 {
7451   \newcommand*{\Glsfmtfullpl}[1]{%
7452     \texorpdfstring
7453       {\Glsxtrtitlefullpl{#1}}%
7454       {\Glsxtrinlinefullplformat{#1}{}}%
7455   }
7456 }
7457 {
7458   \newcommand*{\Glsfmtfullpl}[1]{%
7459     \Glsxtrtitlefullpl{#1}}
7460 }

```

1.8 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

`sariesExtraLang`

```

7461 \newcommand*{\RequireGlossariesExtraLang}[1]{%
7462   \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
7463 }

```

`sariesExtraLang`

```

7464 \newcommand*{\ProvidesGlossariesExtraLang}[1]{%
7465   \ProvidesFile{glossariesxtr-#1.ldf}%
7466 }

```

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.)

```

7467 \@ifpackageloaded{tracklang}
7468 {%
7469   \AnyTrackedLanguages
7470   {%
7471     \ForEachTrackedDialect{\this@dialect}{%
7472       \IfTrackedLanguageFileExists{\this@dialect}%
7473       {glossariesxtr-}% prefix
7474       {.ldf}%
7475     }%

```

```

7476         \RequireGlossariesExtraLang{\CurrentTrackedTag}%
7477     }%
7478     {%
7479     }%
7480 }%
7481 }%
7482 {}%
7483 }
7484 {}

```

Load glossaries-extra-stylemods if required.

```
7485 \@glsxtr@redefstyles
```

and set the style:

```
7486 \@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:

```
7487 \NeedsTeXFormat{LaTeX2e}
7488 \ProvidesPackage{glossaries-extra-stylemods}[2017/04/18 v1.14 (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`

```
7489 \newcommand*{\@glxtr@loadstyles}{%
7490 \DeclareOption*{%
7491   \IfFileExists{glossary-\CurrentOption.sty}%
7492   {\eappto\@glxtr@loadstyles{%
7493     \noexpand\RequirePackage{glossary-\CurrentOption}}}%
7494   {\PackageError{glossaries-extra-styles}%
7495     {Unknown option '\CurrentOption'}{}}
7496 }
```

Process the package options:

```
7497 \ProcessOptions
```

Load the required packages:

```
7498 \@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`

```
7499 \providecommand{\renewglossarystyle}[2]{%
7500   \ifcsundef{@glstyle@#1}%
7501   {%
7502     \PackageError{glossaries}{Glossary style '#1' isn't already defined}{}%
```

```

7503 }%
7504 {%
7505   \csdef{@glsstyle@#1}{#2}%
7506 }%
7507 }

```

2.2 List-Like Styles

The list-like styles mostly already use the post description hook. Only the listdotted style need modifying.

```

7508 \ifcsdef{@glsstyle@listdotted}
7509 {%
7510   \renewglossarystyle{listdotted}{%
7511     \setglossarystyle{list}%
7512     \renewcommand*{\glossentry}[2]{%
7513       \item[]\makebox[\glslistdottedwidth][l]{%
7514         \glstryitem{##1}%
7515         \glstarget{##1}{\glossentryname{##1}}%
7516         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
7517         \glossentrydesc{##1}\glspostdescription}%
7518     \renewcommand*{\subglossentry}[3]{%
7519       \item[]\makebox[\glslistdottedwidth][l]{%
7520         \glssubentryitem{##2}%
7521         \glstarget{##2}{\glossentryname{##2}}%
7522         \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}%
7523         \glossentrydesc{##2}\glspostdescription}%
7524   }
7525 }
7526 {}

```

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.

```

7527 \ifcsdef{@glsstyle@long3col}
7528 {%
7529   \renewglossarystyle{long3col}{%
7530     \renewenvironment{theglossary}%
7531       {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
7532       {\end{longtable}}}%
7533     \renewcommand*{\glossaryheader}{}%
7534     \renewcommand*{\glsgroupheading}[1]{}%
7535     \renewcommand{\glossentry}[2]{%
7536       \glstryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7537       \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline

```

```

7538 }%
7539 \renewcommand{\subglossentry}[3]{%
7540     &
7541     \glssubentryitem{##2}%
7542     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7543     ##3\tabularnewline
7544 }%
7545 \renewcommand*\{ \glsgroupskip }{%
7546     \ifglsgnoglobskip\else & &\tabularnewline\fi}%
7547 }
7548 }
7549 {}

```

Four column style:

```

7550 \ifcsdef{@glsstyle@long4col}
7551 {%
7552     \renewglossarystyle{long4col}{%
7553         \renewenvironment{theglossary}%
7554             {\begin{longtable}{llll}}%
7555             {\end{longtable}}%
7556         \renewcommand*\{ \glossaryheader }{%
7557             \renewcommand*\{ \glsgroupheading }[1]{%
7558                 \renewcommand{\glossentry}[2]{%
7559                     \glssubentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7560                     \glossentrydesc{##1}\glspostdescription &
7561                     \glossentrysymbol{##1} &
7562                     ##2\tabularnewline
7563                 }%
7564                 \renewcommand{\subglossentry}[3]{%
7565                     &
7566                     \glssubentryitem{##2}%
7567                     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7568                     \glossentrysymbol{##2} & ##3\tabularnewline
7569                 }%
7570                 \renewcommand*\{ \glsgroupskip }{%
7571                     \ifglsgnoglobskip\else & &\tabularnewline\fi}%
7572             }
7573         }
7574     }

```

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.

```

7575 \ifcsdef{@glsstyle@longragged3col}
7576 {%
7577     \renewglossarystyle{longragged3col}{%

```

```

7578 \renewenvironment{theglossary}%
7579     {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%
7580      >{\raggedright}p{\glspagelistwidth}}}%
7581     {\end{longtable}}}%
7582 \renewcommand*{\glossaryheader}{}%
7583 \renewcommand*{\glsgroupheading}[1]{}%
7584 \renewcommand{\glossentry}[2]{%
7585     \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7586     \glossentrydesc{##1}\glspostdescription & ##2\tabularnewline
7587 }%
7588 \renewcommand{\subglossentry}[3]{%
7589     &
7590     \glssubentryitem{##2}%
7591     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7592     ##3\tabularnewline
7593 }%
7594 \renewcommand*{\glsgroupskip}{%
7595     \ifglsgnogroupskip\else & \tabularnewline\fi}%
7596 }
7597 }
7598 {}

```

Four column style:

```

7599 \ifcsdef{@glstyle@altlongragged4col}
7600 {%
7601     \renewglossarystyle{altlongragged4col}{%
7602         \renewenvironment{theglossary}%
7603             {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%
7604              >{\raggedright}p{\glspagelistwidth}}}%
7605             {\end{longtable}}}%
7606         \renewcommand*{\glossaryheader}{}%
7607         \renewcommand*{\glsgroupheading}[1]{}%
7608         \renewcommand{\glossentry}[2]{%
7609             \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7610             \glossentrydesc{##1}\glspostdescription & \glossentrysymbol{##1} &
7611             ##2\tabularnewline
7612         }%
7613         \renewcommand{\subglossentry}[3]{%
7614             &
7615             \glssubentryitem{##2}%
7616             \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7617             \glossentrysymbol{##2} & ##3\tabularnewline
7618         }%
7619         \renewcommand*{\glsgroupskip}{%
7620             \ifglsgnogroupskip\else & \tabularnewline\fi}%
7621     }
7622 }
7623 {}

```

2.5 Supertabular Styles

The three and four column styles require adjustment, but not the two column styles.

```
7624 \ifcsdef{@glsstyle@super3col}
7625 {%
7626   \renewglossarystyle{super3col}{%
7627     \renewenvironment{theglossary}%
7628       {\tablehead{}}\tabletail{}}%
7629     \begin{supertabular}{\lp{@glsdescwidth}p{@glspagelistwidth}}}%
7630     {\end{supertabular}}}%
7631   \renewcommand*{\glossaryheader}{}%
7632   \renewcommand*{\glsgroupheading}[1]{}%
7633   \renewcommand{\glossentry}[2]{%
7634     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
7635     \glsentrydesc{##1}\glspostdescription & ##2\tabularnewline
7636   }%
7637   \renewcommand{\subglossentry}[3]{%
7638     &
7639     \glssubentryitem{##2}%
7640     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
7641     ##3\tabularnewline
7642   }%
7643   \renewcommand*{\glsgroupskip}{%
7644     \ifglsnogroupskip\else & &\tabularnewline\fi}%
7645 }
7646 }
7647 {}
```

Four column styles:

```
7648 \ifcsdef{@glsstyle@super4col}
7649 {%
7650   \renewglossarystyle{super4col}{%
7651     \renewenvironment{theglossary}%
7652       {\tablehead{}}\tabletail{}}%
7653     \begin{supertabular}{\llll}}%
7654     {\end{supertabular}}}%
7655   \renewcommand*{\glossaryheader}{}%
7656   \renewcommand*{\glsgroupheading}[1]{}%
7657   \renewcommand{\glossentry}[2]{%
7658     \glstryitem{##1}\glstarget{##1}{\glsentryname{##1}} &
7659     \glsentrydesc{##1}\glspostdescription &
7660     \glsentrysymbol{##1} & ##2\tabularnewline
7661   }%
7662   \renewcommand{\subglossentry}[3]{%
7663     &
7664     \glssubentryitem{##2}%
7665     \glstarget{##2}{\strut}\glsentrydesc{##2}\glspostdescription &
7666     \glsentrysymbol{##2} & ##3\tabularnewline
7667   }%
7668   \renewcommand*{\glsgroupskip}{%
```

```

7669     \ifglsnogroupskip\else & & \tabularnewline\fi}%
7670 }
7671 }
7672 {}

```

2.6 Super Ragged Styles

The three and four column styles require adjustment, but not the two column styles.

```

7673 \ifcsdef{@glsstyle@superragged3col}
7674 {%
7675   \renewglossarystyle{superragged3col}{%
7676     \renewenvironment{theglossary}%
7677       {\tablehead{}\tabletail}%
7678       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}%
7679         >{\raggedright}p{\glspagelistwidth}}}%
7680       {\end{supertabular}}}%
7681   \renewcommand*{\glossaryheader}{}%
7682   \renewcommand*{\glsgroupheading}[1]{}%
7683   \renewcommand{\glossentry}[2]{%
7684     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7685     \glossentrydesc{##1}\glspostdescription &
7686     ##2\tabularnewline
7687   }%
7688   \renewcommand{\subglossentry}[3]{%
7689     &
7690     \glssubentryitem{##2}%
7691     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7692     ##3\tabularnewline
7693   }%
7694   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else &
7695     &\tabularnewline\fi}%
7696 }
7697 }
7698 {}

```

Four columns:

```

7699 \ifcsdef{@glsstyle@altsuperragged4col}
7700 {%
7701   \renewglossarystyle{altsuperragged4col}{%
7702     \renewenvironment{theglossary}%
7703       {\tablehead{}\tabletail}%
7704       \begin{supertabular}{1>{\raggedright}p{\glsdescwidth}l%
7705         >{\raggedright}p{\glspagelistwidth}}}%
7706       {\end{supertabular}}}%
7707   \renewcommand*{\glossaryheader}{}%
7708   \renewcommand{\glossentry}[2]{%
7709     \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
7710     \glossentrydesc{##1}\glspostdescription &
7711     \glossentrysymbol{##1} & ##2\tabularnewline

```

```

7712 }%
7713 \renewcommand{\subglossentry}[3]{%
7714     &
7715     \glssubentryitem{##2}%
7716     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription &
7717     \glossentrysymbol{##2} & ##3\tabularnewline
7718 }%
7719 \renewcommand*{\glsgroupskip}{\ifglsgnোগroupskip\else & &
7720     &\tabularnewline\fi}%
7721 }
7722 }
7723 {}

```

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`.

```

7724 \ifdef{\@glsstyle@inline}
7725 {%
7726     \renewcommand*{\glspostinline}{.\spacefactor\sfcode'\.}
7727     Just use \glxtrpostdescription instead of \glspostdescription.
7728     \renewcommand*{\glsinlinedescformat}[3]{%
7729         \space#1\glxtrpostdescription}
7729     \renewcommand*{\glsinlinesubdescformat}[3]{%
7730         #1\glxtrpostdescription}
7731 }
7732 {}

```

2.8 Tree Styles

The `alttree` style is redefined to make it easier to made minor adjustments.

```

7733 \ifdef{\@glsstyle@alttree}
7734 {%

```

Only redefine this style if it's already been defined.

mbolDescLocation

```
\glxtraltrtreeSymbolDescLocation{\langle label \rangle}{\langle location list \rangle}
```

Layout the symbol, description and location for top-level entries.

```

7735 \newcommand{\glxtraltrtreeSymbolDescLocation}[2]{%
7736     {%
7737         \let\par\glxtrAltTreePar

```

```

7738     \ifglshassymbol{#1}{(\glossentrysymbol{#1})\space}{}%
7739     \glossentrydesc{#1}\glspostdescription \space #2\par
7740 }%
7741 }

```

`\trAltTreeIndent` Paragraph indent for subsequent paragraphs in multi-paragraph descriptions.

```

7742 \newlength\glxstrAltTreeIndent

```

`\lsxtrAltTreePar` Multi-paragraph descriptions need to keep the hanging indent.

```

7743 \newcommand{\glxstrAltTreePar}{%
7744   @@par
7745   \glxstrAltTreeSetHangIndent
7746   \setlength{\parindent}{\dimexpr\hangindent+\glxstrAltTreeIndent}%
7747 }

```

`\symbolDescLocation` `\glxstralttreeSubSymbolDescLocation{<level>}{<label>}{<location list>}`

Layout the symbol, description and location for sub-entries. Defaults to the same as the top-level.

```

7748 \newcommand{\glxstralttreeSubSymbolDescLocation}[3]{%
7749   \glxstralttreeSymbolDescLocation{#2}{#3}%
7750 }

```

`\trtreeTopindent` The original style has to keep computing the width of the name at each entry. This register allows the style to compute it once for the top-level at the start of the glossary.

```

7751 \newlength\glxstrtreeTopindent

```

`\glxstralttreeInit` User-level initialisation for the alttree style.

```

7752 \newcommand*{\glxstralttreeInit}{%
7753   \settowidth{\glxstrtreeTopindent}{\glstreenamfmt{\glsgetwidestname\space}}%
7754   \glxstrAltTreeIndent=\parindent
7755 }

```

`\eglissetwidest` The original `\glissetwidest` only uses `\def`. This uses `\protected@csedef`.

```

7756 \newcommand*{\eglissetwidest}[2][0]{%
7757   \protected@csedef{@glswidestname\romannumeral#1}{#2}%
7758 }

```

`\xglissetwidest` Like the above but uses `\protected@csxdef`.

```

7759 \newcommand*{\xglissetwidest}[2][0]{%
7760   \protected@csxdef{@glswidestname\romannumeral#1}{#2}%
7761 }

```

`\lsgetwidestname` Provide a user-level macro to obtain the widest top-level name.

```

7762 \newcommand*{\glsggetwidestname}{\@glswidestname}

```


`etwidestsubname` Provide a user-level macro to obtain the widest sub-entry name.

```

7763 \newcommand*\glsgetwidestsubname}[1]{%
7764   \ifcsundef{@glswidestname\romannumeral#1}%
7765     {\@glswidestname}%
7766     {\csuse{@glswidestname\romannumeral#1}}%
7767 }

```

`estTopLevelName` CamelCase is easier for long command names. Provide a CamelCase synonym of `\glsfindwidesttoplevelname`

```

7768 \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.

```

7769 \newrobustcmd*\glsFindWidestUsedTopLevelName}[1][\@glo@types]{%
7770   \dimen@=0pt\relax
7771   \gls@tmplen=0pt\relax
7772   \forallglossaries[#1]{\@gls@type}%
7773   {%
7774     \forglsentries[\@gls@type]{\@glo@label}%
7775     {%
7776       \ifglsused{\@glo@label}%
7777       {%
7778         \ifglshasparent{\@glo@label}%
7779         {}%
7780         {%
7781           \settowidth{\dimen@}%
7782             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
7783           \ifdim\dimen@>\gls@tmplen
7784             \gls@tmplen=\dimen@
7785             \eglssetwidest{\glsentryname{\@glo@label}}%
7786           \fi
7787         }%
7788       }%
7789     }%
7790   }%
7791 }%
7792 }

```

`destUsedAnyName` Like the above but doesn't check the parent key. Useful if all levels should have the same width for the name.

```

7793 \newrobustcmd*\glsFindWidestUsedAnyName}[1][\@glo@types]{%
7794   \dimen@=0pt\relax
7795   \gls@tmplen=0pt\relax
7796   \forallglossaries[#1]{\@gls@type}%
7797   {%
7798     \forglsentries[\@gls@type]{\@glo@label}%
7799     {%

```

```

7800     \ifglused{\@glo@label}%
7801     {%
7802         \settowidth{\dimen@}%
7803         {\glstreenamfmt{\glentryname{\@glo@label}}}%
7804         \ifdim\dimen@>\gls@tmplen
7805             \gls@tmplen=\dimen@
7806             \eglssetwidest{\glentryname{\@glo@label}}%
7807         \fi
7808     }%
7809     {%
7810 }%
7811 }%
7812 }

```

ndWidestAnyName Like the above but doesn't check if the entry has been used.

```

7813 \newrobustcmd*{\glsFindWidestAnyName}[1][\@glo@types]{%
7814     \dimen@=0pt\relax
7815     \gls@tmplen=0pt\relax
7816     \forallglossaries[#1]{\@gls@type}%
7817     {%
7818         \forglsentries[\@gls@type]{\@glo@label}%
7819         {%
7820             \settowidth{\dimen@}%
7821             {\glstreenamfmt{\glentryname{\@glo@label}}}%
7822             \ifdim\dimen@>\gls@tmplen
7823                 \gls@tmplen=\dimen@
7824                 \eglssetwidest{\glentryname{\@glo@label}}%
7825             \fi
7826         }%
7827     }%
7828 }

```

estUsedLevelTwo This is like `\glsFindWidestUsedTopLevelName` but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

```

7829 \newrobustcmd*{\glsFindWidestUsedLevelTwo}[1][\@glo@types]{%
7830     \dimen@=0pt\relax
7831     \dimen@i=0pt\relax
7832     \dimen@ii=0pt\relax
7833     \forallglossaries[#1]{\@gls@type}%
7834     {%
7835         \forglsentries[\@gls@type]{\@glo@label}%
7836         {%
7837             \ifglused{\@glo@label}%
7838             {%
7839                 \ifglshasparent{\@glo@label}%
7840                 {%
7841                     \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
7842                     \ifglshasparent{\@glo@parent}%
7843                 }%

```

```

7844      \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
7845      \ifglshasparent{\@glo@parent}%
7846      {%
7847      {%
7848          \settowidth{\gls@tmplen}%
7849          {\glstreenamefmt{\glsentryname{\@glo@label}}}%
7850          \ifdim\gls@tmplen>\dimen@ii
7851          \dimen@ii=\gls@tmplen
7852          \eglssetwidest[2]{\glsentryname{\@glo@label}}%
7853          \fi
7854      }%
7855      }%
7856      {%
7857          \settowidth{\gls@tmplen}%
7858          {\glstreenamefmt{\glsentryname{\@glo@label}}}%
7859          \ifdim\gls@tmplen>\dimen@i
7860          \dimen@i=\gls@tmplen
7861          \eglssetwidest[1]{\glsentryname{\@glo@label}}%
7862          \fi
7863      }%
7864      }%
7865      {%
7866          \settowidth{\gls@tmplen}%
7867          {\glstreenamefmt{\glsentryname{\@glo@label}}}%
7868          \ifdim\gls@tmplen>\dimen@
7869          \dimen@=\gls@tmplen
7870          \eglssetwidest{\glsentryname{\@glo@label}}%
7871          \fi
7872      }%
7873      }%
7874      {}%
7875      }%
7876      }%
7877      }

```

`dWidestLevelTwo` This is like `\glsFindWidestUsedLevelTwo` but doesn't check if the entry has been used.

```

7878      \newrobustcmd*{\glsFindWidestLevelTwo}[1][\@glo@types]{%
7879          \dimen@=0pt\relax
7880          \dimen@i=0pt\relax
7881          \dimen@ii=0pt\relax
7882          \foralllglossaries[#1]{\@gls@type}%
7883          {%
7884              \forglsentries[\@gls@type]{\@glo@label}%
7885              {%
7886                  \ifglshasparent{\@glo@label}%
7887                  {%
7888                      \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@label}@parent}}%
7889                      \ifglshasparent{\@glo@parent}%
7890                      {%

```

```

7891      \edef\@glo@parent{\csuse{glo@\glsdetoklabel{\@glo@parent}@parent}}%
7892      \ifglshasparent{\@glo@parent}%
7893      {%
7894      {%
7895          \settowidth{\gls@tmplen}%
7896              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
7897          \ifdim\gls@tmplen>\dimen@ii
7898              \dimen@ii=\gls@tmplen
7899              \eglssetwidest[2]{\glsentryname{\@glo@label}}%
7900          \fi
7901      }%
7902      }%
7903      {%
7904          \settowidth{\gls@tmplen}%
7905              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
7906          \ifdim\gls@tmplen>\dimen@i
7907              \dimen@i=\gls@tmplen
7908              \eglssetwidest[1]{\glsentryname{\@glo@label}}%
7909          \fi
7910      }%
7911      }%
7912      {%
7913          \settowidth{\gls@tmplen}%
7914              {\glstreenamefmt{\glsentryname{\@glo@label}}}%
7915          \ifdim\gls@tmplen>\dimen@
7916              \dimen@=\gls@tmplen
7917              \eglssetwidest{\glsentryname{\@glo@label}}%
7918          \fi
7919      }%
7920      }%
7921      }%
7922      }

```

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.

```

7923 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbol}[2][\@glo@types]{%
7924     \dimen@=0pt\relax
7925     \gls@tmplen=0pt\relax
7926     #2=0pt\relax
7927     \forallglossaries[#1]{\@gls@type}%
7928     {%
7929         \forglsentries[\@gls@type]{\@glo@label}%
7930         {%
7931             \ifglused{\@glo@label}%
7932             {%
7933                 \settowidth{\dimen@}%
7934                     {\glstreenamefmt{\glsentryname{\@glo@label}}}%
7935                 \ifdim\dimen@>\gls@tmplen
7936                     \gls@tmplen=\dimen@

```

```

7937         \eglssetwidest{\glsentryname{\@glo@label}}%
7938     \fi
7939     \settowidth{\dimen@}%
7940     {\glsentrysymbol{\@glo@label}}%
7941     \ifdim\dimen@>#2\relax
7942         #2=\dimen@
7943     \fi
7944 }%
7945 {}%
7946 }%
7947 }%
7948 }

```

stAnyNameSymbol Like the above but doesn't check if the entry has been used.

```

7949 \newrobustcmd*{\glsFindWidestAnyNameSymbol}[2][\@glo@types]{%
7950     \dimen@=0pt\relax
7951     \gls@tmplen=0pt\relax
7952     #2=0pt\relax
7953     \forallglossaries[#1]{\@gls@type}%
7954     {%
7955         \forglsentries[\@gls@type]{\@glo@label}%
7956         {%
7957             \settowidth{\dimen@}%
7958             {\glstreenamfmt{\glsentryname{\@glo@label}}}%
7959             \ifdim\dimen@>\gls@tmplen
7960                 \gls@tmplen=\dimen@
7961                 \eglssetwidest{\glsentryname{\@glo@label}}%
7962             \fi
7963             \settowidth{\dimen@}%
7964             {\glsentrysymbol{\@glo@label}}%
7965             \ifdim\dimen@>#2\relax
7966                 #2=\dimen@
7967             \fi
7968         }%
7969     }%
7970 }

```

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.

```

7971 \newrobustcmd*{\glsFindWidestUsedAnyNameSymbolLocation}[3][\@glo@types]{%
7972     \dimen@=0pt\relax
7973     \gls@tmplen=0pt\relax
7974     #2=0pt\relax
7975     #3=0pt\relax
7976     \forallglossaries[#1]{\@gls@type}%
7977     {%
7978         \forglsentries[\@gls@type]{\@glo@label}%

```

```

7979     {%
7980         \ifglsused{\@glo@label}%
7981     {%
7982         \settowidth{\dimen@}%
7983             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
7984         \ifdim\dimen@>\gls@tmplen
7985             \gls@tmplen=\dimen@
7986             \eglssetwidest{\glsentryname{\@glo@label}}%
7987         \fi
7988         \settowidth{\dimen@}%
7989             {\glsentrysymbol{\@glo@label}}%
7990         \ifdim\dimen@>#2\relax
7991             #2=\dimen@
7992         \fi
7993         \settowidth{\dimen@}%
7994             {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
7995         \ifdim\dimen@>#3\relax
7996             #3=\dimen@
7997         \fi
7998     }%
7999 }%
8000 }%
8001 }%
8002 }

```

eSymbolLocation Like the `\glsFindWidestUsedAnyNameSymbol` but doesn't check if the entry has been used.

```

8003 \newrobustcmd*{\glsFindWidestAnyNameSymbolLocation}[3][\@glo@types]{%
8004     \dimen@=0pt\relax
8005     \gls@tmplen=0pt\relax
8006     #2=0pt\relax
8007     #3=0pt\relax
8008     \foralllglossaries[#1]{\@gls@type}%
8009     {%
8010         \forglsentries[\@gls@type]{\@glo@label}%
8011         {%
8012             \settowidth{\dimen@}%
8013                 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8014             \ifdim\dimen@>\gls@tmplen
8015                 \gls@tmplen=\dimen@
8016                 \eglssetwidest{\glsentryname{\@glo@label}}%
8017             \fi
8018             \settowidth{\dimen@}%
8019                 {\glsentrysymbol{\@glo@label}}%
8020             \ifdim\dimen@>#2\relax
8021                 #2=\dimen@
8022             \fi
8023             \settowidth{\dimen@}%
8024                 {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
8025             \ifdim\dimen@>#3\relax

```

```

8026         #3=\dimen@
8027     \fi
8028 }%
8029 }%
8030 }

```

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.

```

8031 \newrobustcmd*{\glsFindWidestUsedAnyNameLocation}[2][\@glo@types]{%
8032     \dimen@=0pt\relax
8033     \gls@tmplen=0pt\relax
8034     #2=0pt\relax
8035     \forallglossaries[#1]{\@gls@type}%
8036     {%
8037         \forallglsentries[\@gls@type]{\@glo@label}%
8038         {%
8039             \ifglsused{\@glo@label}%
8040             {%
8041                 \settowidth{\dimen@}%
8042                 {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8043                 \ifdim\dimen@>\gls@tmplen
8044                     \gls@tmplen=\dimen@
8045                     \eglssetwidest{\glsentryname{\@glo@label}}%
8046                 \fi
8047                 \settowidth{\dimen@}%
8048                 {\GlsXtrFormatLocationList{\glsentrynumberlist{\@glo@label}}}%
8049                 \ifdim\dimen@>#2\relax
8050                     #2=\dimen@
8051                 \fi
8052             }%
8053         }%
8054     }%
8055 }%
8056 }

```

AnyNameLocation Like the `\glsFindWidestAnyNameLocation` but doesn't check the **first use** flag.

```

8057 \newrobustcmd*{\glsFindWidestAnyNameLocation}[2][\@glo@types]{%
8058     \dimen@=0pt\relax
8059     \gls@tmplen=0pt\relax
8060     #2=0pt\relax
8061     \forallglossaries[#1]{\@gls@type}%
8062     {%
8063         \forallglsentries[\@gls@type]{\@glo@label}%
8064         {%
8065             \settowidth{\dimen@}%
8066             {\glstreenamefmt{\glsentryname{\@glo@label}}}%
8067             \ifdim\dimen@>\gls@tmplen
8068                 \gls@tmplen=\dimen@

```

```

8069         \eglssetwidest{\glstryname{\@glo@label}}}%
8070     \fi
8071     \settowidth{\dimen@}%
8072         {\GlsXtrFormatLocationList{\glstrynumberlist{\@glo@label}}}%
8073     \ifdim\dimen@>#2\relax
8074         #2=\dimen@
8075     \fi
8076 }%
8077 }%
8078 }

```

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`.

```

8079 \newcommand*{\glsxtrComputeTreeIndent}[1]{%
8080     \glstreeindent=\glsxtrtreetopindent\relax
8081 }

```

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.

```

8082 \newcommand*{\glsxtrComputeTreeSubIndent}[3]{%
8083     \ifcsundef{@glswidestname\romannumeral#1}%
8084     {%
8085         \settowidth{#3}{\glstreenamfmt{\@glswidestname\space}}%
8086     }%
8087     {%
8088         \settowidth{#3}{\glstreenamfmt{%
8089             \csname @glswidestname\romannumeral#1\endcsname\space}}%
8090     }%
8091 }

```

treeSetHangIndent Set `\hangindent` for top-level entries:

```

8092 \newcommand*{\glsxtrAltTreeSetHangIndent}{\hangindent\glstreeindent}

```

treeSubHangIndent Set `\hangindent` for sub-entries:

```

8093 \newcommand*{\glsxtrAltTreeSetSubHangIndent}[1]{\hangindent\glstreeindent}

```

Redefine `alttree`:

```

8094 \renewglossarystyle{alttree}{%
8095     \renewenvironment{theglossary}%
8096     {%
8097         \glsxtralttreeInit

```



```

8098     \def\@gls@prevlevel{-1}%
8099     \mbox{}\par}%
8100     {\par}%
8101 \renewcommand*\glossaryheader{}%
8102 \renewcommand*\glsgroupheading[1]{}%
8103 \renewcommand\glossentry[2]{%
8104     \ifnum\@gls@prevlevel=0\relax
8105     \else
8106         \glstrComputeTreeIndent{##1}%
8107     \fi
8108     \parindent\glstreeindent
8109     \glstrAltTreeSetHangIndent
8110     \makebox[Opt][r]%
8111     {%
8112         \glstreenamebox{\glstreeindent}%
8113         {%
8114             \glstryitem{##1}%
8115             \glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
8116         }%
8117     }%
8118     \glstralttreeSymbolDescLocation{##1}{##2}%
8119     \def\@gls@prevlevel{0}%
8120 }
8121 \renewcommand\subglossentry[3]{%
8122     \ifnum##1=1\relax
8123         \glssubentryitem{##2}%
8124     \fi
8125     \ifnum\@gls@prevlevel=##1\relax
8126     \else
8127         \glstrComputeTreeSubIndent{##1}{##2}{\@gls@tmplen}%
8128         \ifnum\@gls@prevlevel<##1\relax
8129             \setlength\glstreeindent\@gls@tmplen
8130             \addtolength\glstreeindent\parindent
8131             \parindent\glstreeindent
8132         \else
8133             \ifnum\@gls@prevlevel=0\relax
8134                 \glstrComputeTreeIndent{##2}%
8135             \else
8136                 \glstrComputeTreeSubIndent{\@gls@prevlevel}{##2}{\glstreeindent}%
8137             \fi
8138             \addtolength\parindent{-\glstreeindent}%
8139             \setlength\glstreeindent\parindent
8140         \fi
8141     \fi
8142     \glstrAltTreeSetSubHangIndent{##1}%
8143     \makebox[Opt][r]{\glstreenamebox{\@gls@tmplen}{%
8144         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%
8145     \glstralttreeSubSymbolDescLocation{##1}{##2}{##3}%
8146     \def\@gls@prevlevel{##1}%

```

```

8147     }%
8148     \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8149   }
8150 }%
8151 {%

```

Assume the style isn't required if it hasn't already been defined.

```

8152 }

```

Reset the default style

```

8153 \ifx\@glossary@default@style\relax
8154 \else
8155   \setglossarystyle{\@glxtr@current@style}
8156 \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.

Change History

0.1 (2015-11-22)	
General: Initial experimental release	4
0.2 (2015-11-30)	
\Glsfmtshort: new	210
\glsfmtshort: new	209
\Glsfmtshortpl: new	210
\glsfmtshortpl: new	209
short: switched inline full form to short (long)	172
0.3 (2015-12-02)	
\@ACRlong: added redefinition	55
\@ACRlongpl: added redefinition	56
\@ACRshort: added redefinition	53
\@ACRshortpl: added redefinition	54
\@Acrlong: added redefinition	54
\@Acrlongpl: added redefinition	55
\@Acrshort: added redefinition	52
\@Acrshortpl: added redefinition	53
\@GLSdesc: added redefinition	48
\@GLSdescplural@: added redefinition	49
\@GLSfirst@: added redefinition	46
\@GLSfirstplural@: added redefinition	47
\@GLSname@: added redefinition	48
\@GLSplural@: added redefinition	46
\@GLSSymbol@: added redefinition	49
\@GLSSymbolplural@: added redefinition	50
\@GLStext@: added redefinition	45
\@GLSuseri@: added redefinition	50
\@GLSuserii@: added redefinition	50
\@GLSuseriii@: added redefinition	51
\@GLSuseriv@: added redefinition	51
\@GLSuserv@: added redefinition	51
\@GLSuservi@: added redefinition	52
\@GLSdesc@: added redefinition	48
\@GLSdescplural@: added redefinition	48
\@GLSfirst@: added redefinition	46
\@GLSfirstplural@: added redefinition	47
\@GLSname@: added redefinition	48
\@GLSplural@: added redefinition	46
\@GLSSymbol@: added redefinition	49
\@GLSSymbolplural@: added redefinition	49
\@GLStext@: added redefinition	45
\@GLSuseri@: added redefinition	50
\@GLSuserii@: added redefinition	50
\@GLSuseriii@: added redefinition	51
\@GLSuseriv@: added redefinition	51
\@GLSuserv@: added redefinition	51
\@GLSuservi@: added redefinition	51
\@acrlong: added redefinition	54
\@acrlongpl: added redefinition	55
\@acrshort: added redefinition	52
\@acrshortpl: added redefinition	53
\@gls@field@link: added optional argument	40
\@glsdescplural@: added redefinition	48
\@glsfirst@: added redefinition	45
\@glsfirstplural@: added redefinition	47
\@glsplural@: added redefinition	46
\@glssymbolplural@: added redefinition	49
\@glsxtr@defaultnoglossarywarning: new	102
\@glsxtr@field@linkdefs: new	44
\@glsxtr@insertdots: new	144
\@print@glossary: added redefinition	99
\glsabbrvdefaultfont: renamed from \abbrvdefaultfont	148
\glsaccessdesc: new	114
\glsaccessdescplural: new	115
\glsaccessfirst: new	112
\glsaccessfirstplural: new	112
\glsaccesslong: new	117
\glsaccesslong: new	116
\glsaccessname: new	110
\glsaccessplural: new	111
\glsaccessshort: new	115
\glsaccessshort: new	115
\glsaccessshortpl: new	116

\glsaccessshortpl: new	116	\cGLSpl: new	78
\glsaccesssymbol: new	113	\cGLSpl@: new	79
\glsaccesssymbolplural: new	113	\glsxtr@setentrycountunsetattr:	
\glsaccesstext: new	111	new	74
\glstentryfmt: added check for short ..	39	\cGLS: new	78
\glslongpltok: new	144	\cGLSformat: new	78
\glsshortpltok: new	144	\cGLSpl: new	78
\glsxtr@newabbreviation: fixed family		\cGLSplformat: new	79
name in \setkeys	145	\GlossariesExtraWarningNoLine:	
\glsxtrdiscardperiod: added check		new	11
for plural	141	\glsenableentrycount: new	74
\GLSxtrlongpl: new	158	\glsfirstabbrvdefaultfont: new ..	148
\Glsxtrlongpl: new	157	\glsfirstlongdefaultfont: new ...	148
\glsxtrlongpl: new	157	\Glsfmtfirst: new	212
\glsxtrNoGlossaryWarning: new	14	\glsfmtfirst: new	211
\glsxtrpostlinkAddDescOnFirstUse:		\Glsfmtfirstpl: new	212
new	141	\glsfmtfirstpl: new	212
\glsxtrpostlinkAddSymbolOnFirstUse:		\Glsfmtplural: new	211
new	141	\glsfmtplural: new	211
\glsxtrpostlinkendsentence: new ..	140	\Glsfmtshort: changed to use	
\GLSxtrshortpl: new	156	\Glsxtrtitleshort	210
\Glsxtrshortpl: new	155	renamed from \glstentryfmtshort ..	210
\glsxtrshortpl: new	155	\glsfmtshort: changed to use	
short-long-desc: fixed name to use		\glsxtrtitleshort	209
\glslabeltok	167	renamed from \glstentryfmtshort ..	209
long-short-desc: fixed name to use		\Glsfmtshortpl: changed to use	
\glslabeltok	165	\Glsxtrtitleshortpl	210
0.4 (2015-12-03)		renamed from	
\@glsxtr@doabbreviationsdef: added		\Glstentryfmtshortpl	210
redefinition of \acronymtype	12	\glsfmtshortpl: changed to use	
\Glsfmtshort: changed to use		\glsxtrtitleshortpl	209
\Glsxtrshort	210	renamed from	
\glsfmtshort: changed to use		\glstentryfmtshortpl	209
\glsxtrshort	209	\Glsfmttext: new	211
\Glsfmtshortpl: changed to use		\glsfmttext: new	210
\glsxtrshortpl	210	\glshasattribute: new	122
\glsfmtshortpl: changed to use		\glshascategoryattribute: new ...	121
\glsxtrshortpl	209	\GlsXtrEnableEntryCounting: new ..	73
\glsxtrifemptyglossary: new	17	\glsxtrifcounttrigger: new	76
\glsxtrnewnumber: added extra		\glsxtrscfont: new	176
argument	125	\glsxtrscsuffix: new	176
\glsxtrnewsymbol: added extra		\glsxtrsmfont: new	180
argument	125	\glsxtrsmsuffix: new	180
\MakeAcronymsAbbreviations: set the		short-em: new	187
default type to \acronymtype	87	short-em-desc: new	187
\newterm: fixed name argument	124	short-em-footnote: new	189
0.5 (2015-12-07)		short-em-long: new	185
\cGLS: new	78	short-em-long-desc: new	186
\cGLS@: new	78	short-em-postfootnote: new	189

short-sc-footnote: new	179	\Glsxtrheadtext: now uses headuc	
short-sc-postfootnote: new	180	attribute	203
short-sm: new	181	\glxtrheadtext: now uses headuc	
short-sm-desc: new	182	attribute	202
short-sm-footnote: new	183	short-long: switch off regular attribute	
short-sm-long: new	181	if set	166
short-sm-long-desc: new	181	short-long-desc: switch off regular	
short-sm-postfootnote: new	183	attribute if set	167
long-noshort-em: new	188	long-short: switch off regular attribute	
long-noshort-em-desc: new	188	if set	164
long-noshort-sm: new	182	long-short-desc: switch off regular	
long-noshort-sm-desc: new	182	attribute if set	165
long-short-em: new	184	footnote: switch off regular attribute if	
long-short-em-desc: new	184	set	168
long-short-sm: new	180	postfootnote: switch off regular	
long-short-sm-desc: new	181	attribute if set	170
0.5.1 (2015-12-02)		0.5.2 (2015-12-08)	
\Glsaccessstext: new	111	\@GLSdesc@: added accessibility support	48
0.5.1 (2015-12-07)		\@GLSdescplural@: added accessibility	
\@glxtr@doaccsupp: new	14	support	49
General: removed \ifglxtruseuhead	201	\@GLSfirst@: added accessibility	
\Glsaccessdesc: new	114	support	46
\Glsaccessdescplural: new	115	\@GLSfirstplural@: added accessibility	
\Glsaccessfirst: new	112	support	47
\Glsaccessfirstplural: new	113	\@GLSname@: added accessibility support	48
\Glsaccessname: new	110	\@GLSplural@: added accessibility	
\Glsaccessplural: new	111	support	46
\Glsaccesssymbol: new	113	\@GLSsymbol@: added accessibility	
\Glsaccesssymbolplural: new	114	support	49
\Glsxtrheadfirst: now uses headuc		\@GLSsymbolplural@: added	
attribute	204	accessibility support	50
\glxtrheadfirst: now uses headuc		\@GLStext@: added accessibility support	45
attribute	204	\@GLSdesc@: added accessibility support	48
\Glsxtrheadfirstplural: now uses		\@GLSdescplural@: added accessibility	
headuc attribute	205	support	48
\glxtrheadfirstplural: now uses		\@GLSfirst@: added accessibility	
headuc attribute	205	support	46
\Glsxtrheadplural: now uses headuc		\@GLSfirstplural@: added accessibility	
attribute	204	support	47
\glxtrheadplural: now uses headuc		\@GLSname@: add accessibility support ..	48
attribute	203	\@GLSplural@: added accessibility	
\Glsxtrheadshort: now uses headuc		support	46
attribute	202	\@GLSsymbol@: added accessibility	
\glxtrheadshort: now uses headuc		support	49
attribute	201	\@GLSsymbolplural@: added	
\Glsxtrheadshortpl: now uses headuc		accessibility support	50
attribute	202	\@GLStext@: added accessibility support	45
\glxtrheadshortpl: now uses headuc		\@GLSdesc@: added accessibility support	48
attribute	201		

\@glsdescplural@: added accessibility support	48	\glsxtrnewabbrevpresetkeyhook: new	146
\@glsfirst@: added accessibility support	45	\glsxtrtagfont: new	139
\@glsfirstplural@: added accessibility support	47	\KV@printgloss@nonumberlist: added	39
\@glsname@: added accessibility support	47	\mfu@checkword@do: added	138
\@glsplural@: added accessibility support	46	\setabbreviationstyle: added check for post-definition style switch	161
\@glsymbol@: added accessibility support	49	0.5.3 (2015-12-09)	
\@glsymbolplural@: added accessibility support	49	\@glsxtr@autoindex@at: new	134
\@glstext@: added accessibility support	45	\@glsxtr@autoindex@encap: new ...	135
\@glsxtr@activate@initialtagging: new	139	\@glsxtr@autoindex@esc: new	135
\@glsxtr@do@titlecaps@warn: new	138	\@glsxtr@autoindex@level: new ...	135
\@glsxtr@tag: new	139	\@glsxtr@autoindex@setname: new	133
General: fixed typo in glossaries-accsupp and tidied up code to use just one		\@glsxtr@doabbreviationsdef: new	12
\@ifpackageloaded	110	General: removed	
removed \glsxtrabbrvfmt	158	\GlsXtrNoGlsWarningNoAutoMakeMain	101
\glossaryentrynumbers: added	37	\glsdescwidth: added	36
\Glossentrydesc: added	136	\glspagelistwidth: added	37
\Glossentryname: added	131	\glsxtrdoautoindexname: new	133
\Glossentrysymbol: added	137	\glsxtrpostnamehook: new	132
\glossentrysymbol: added	137	\if@glsxtr@format@override: new	132
\GLSaccessdesc: new	114, 119	\ProvidesGlossariesExtraLang: new	215
\GLSaccessdescplural: new ...	115, 119	\RequireGlossariesExtraLang: new	215
\GLSaccessfirst: new	112, 118	0.5.4 (2015-12-15)	
\GLSaccessfirstplural: new ..	113, 118	\@newglossaryentry@defunitcounters: new	79
\GLSaccesslong: new	117, 120	\@GLSxtr@p@acrlong@: new	67
\GLSaccesslongpl: new	117, 120	\@GLSxtr@p@acrlongpl@: new	67
\Glsaccesslongpl: new	117	\@GLSxtr@p@acrshort@: new	66
\glsaccesslongpl: new	117	\@GLSxtr@p@acrshortpl@: new	66
\GLSaccessname: new	110, 118	\@GLSxtr@p@long@: new	66
\GLSaccessplural: new	112, 118	\@GLSxtr@p@longpl@: new	66
\GLSaccessshort: new	116, 120	\@GLSxtr@p@plural@: new	65
\GLSaccessshortpl: new	116, 120	\@GLSxtr@p@short@: new	65
\GLSaccesssymbol: new	113, 119	\@GLSxtr@p@shortpl@: new	66
\GLSaccesssymbolplural: new	114, 119	\@GLSxtr@p@text@: new	64
\GLSaccessstext: new	111, 118	\@GlsXtrEnableOnTheFly: new	33
\glsentryfmt: moved		\@Glsxtr: new	33
\glssetabbrvfmt from		\@Glsxtr@p@acrlong@: new	67
\glsxtrabbrvfmt to here	39	\@Glsxtr@p@acrlongpl@: new	67
\GlsXtrEnableInitialTagging: new	137	\@Glsxtr@p@acrshort@: new	66
\glsxtrfieldtitlecase: new	126	\@Glsxtr@p@acrshortpl@: new	66
\GlsXtrFormatLocationList: new ...	37	\@Glsxtr@p@long@: new	66
		\@Glsxtr@p@longpl@: new	66
		\@Glsxtr@p@plural@: new	65
		\@Glsxtr@p@short@: new	65
		\@Glsxtr@p@shortpl@: new	65
		\@Glsxtr@p@text@: new	64

\@Glsxtrpl: new	34	\glxtr: new	33
\@alt@glshyp@opt: new	61	\glxtrcat: new	33
\@glscalt@hyp@opt: new	61	\glxtrdowrglossaryhook: new	61
\@glscalt@hyp@opt@char: new	62	\GlsXtrEnableEntryUnitCounting:	
\@glscalt@hyp@opt@keys: new	62	new	85
\@glsc@increment@currunitcount:		\GlsXtrEnableOnTheFly: new	32
new	80	\Glsxtrpl: new	34
\@glsc@local@increment@currunitcount:		\glxtrpl: new	34
new	81	\glxtrpostlocalreset: new	73
\@glsc@setdefault@glslink@opts:		\glxtrpostlocalunset: new	73
new	59	\glxtrpostreset: new	73
\@glxtr: new	33	\glxtrpostunset: new	72
\@glxtr@addunitcounter: new	80	\glxtrprotectlinks: new	64
\@glxtr@currunitcount: new	81	\GlsXtrSetAltModifier: new	62
\@glxtr@ifunitcounter: new	80	\GlsXtrSetDefaultGlsOpts: new	60
\@glxtr@p@acrlong@: new	67	\glxtrstarflywarn: new	33
\@glxtr@p@acrlongpl@: new	67	\GlsXtrWarning: new	34
\@glxtr@p@acrshort@: new	66	\MakeAcronymsAbbreviations: now	
\@glxtr@p@acrshortpl@: new	66	disables \setacronymstyle	87
\@glxtr@p@long@: new	66	1.0 (2016-01-24)	
\@glxtr@p@longpl@: new	66	\@glxtr@autoindexcrossrefs: new .	11
\@glxtr@p@plural@: new	65	\@glxtr@idx@displaynumberlist:	
\@glxtr@p@short@: new	65	new	93
\@glxtr@p@shortpl@: new	65	\@glxtr@idx@entrynumberlist: new	95
\@glxtr@p@text@: new	64	\@glxtr@noidx@displaynumberlist:	
\@glxtr@prevunitcount: new	81	new	94
\@glxtr@setentryunitcountunsetattr:		\@glxtr@noidx@entrynumberlist:	
new	85	new	95
\@glxtr@unitcountlist: new	80	\@glxtr@noidx@numberlistloop:	
\@glxtrpl: new	34	new	94
\@newglossaryentryposthook: added		\@glxtr@reg@glosslist: new	88
empty see value if not set and added		\makeglossaries: new	88
‘see’ to field key map	29	1.01 (2016-02-02)	
\@sGlsXtrEnableOnTheFly: new	32	\glxtrdiscardperiod: added check	
\cGlsformat: added	79	for first use	141
\cGlsformat: added	79	short-desc: fixed typo in	
\cGlsplformat: added	79	\glxtrinlinefullformat and	
\cGlsplformat: added	79	added missing second argument ...	173
\glstdisablehyper: added	63	1.02 (2016-04-25)	
\glstdohyperlink: added	62	\@glxtr@current@style: new	35
\glstdonohyperlink: added	63	\Glsfmtfull: new	214
\glsenableentryunitcount: new	82	\glsfmtfull: new	214
\glshasattribute: added check for		\Glsfmtfullpl: new	215
entry’s existence	122	\glsfmtfullpl: new	214
\glusifattribute: added check for		\Glsfmtlong: new	213
entry’s existence	122	\glsfmtlong: new	213
\glspostlinkhook: added existence		\Glsfmtlongpl: new	213
check	140	\glsfmtlongpl: new	213
\Glsxtr: new	33	\Glsxtrheadfull: new	208

\glxtrheadfull: new	207	\@GLSplural@: set abbreviation and regular format	46
\Glsxtrheadfullpl: new	209	\@GLSsymbol@: set regular format	49
\glxtrheadfullpl: new	208	\@GLSsymbolplural@: set regular format	50
\glxtrheadlong: new	207	\@GLStext@: set abbreviation and regular format	45
\glxtrheadlongpl: new	207	\@GLSuseri@: set regular format	50
\glxtrheadlongpl: new	206	\@GLSuserii@: set regular format	50
\Glsxtrtitlefull: new	209	\@GLSuseriii@: set regular format	51
\glxtrtitlefull: new	208	\@GLSuseriv@: set regular format	51
\Glsxtrtitlefullpl: new	209	\@GLSuseriv@: set regular format	51
\glxtrtitlefullpl: new	208	\@GLSuservi@: set regular format	52
\Glsxtrtitlelong: new	207	\@Glsdesc@: set abbreviation and regular format	48
\glxtrtitlelong: new	206	\@Glsdescplural@: set abbreviation and regular format	48
\Glsxtrtitlelongpl: new	207	\@Glsfirst@: set abbreviation and regular format	46
\glxtrtitlelongpl: new	206	\@Glsfirstplural@: set abbreviation and regular format	47
\ifglxtrinsetinside: new	164	\@Glsname@: set abbreviation and regular format	48
postfootnote: added redef of \glxtrsetupfulldefs	170	\@Glsplural@: set abbreviation and regular format	46
stylemods: new	15	\@Glsymbol@: set regular format	49
1.03 (2016-04-27)		\@Glsymbolplural@: set regular format	50
\@GLSfirstplural@: bug fix: misspelt cs name	47	\@Glstext@: set abbreviation and regular format	45
\@GLSplural@: fixed bug \@GLSplural@ should be redefined not \@GLSplural	46	\@GLSuseri@: set regular format	50
\@Glsfirstplural@: bug fix: misspelt cs name	47	\@GLSuserii@: set regular format	50
\@Glsplural@: fixed bug \@Glsplural@ should be redefined not \@Glsplural	46	\@GLSuseriii@: set regular format	51
\@glsplural@: fixed bug \@glsplural@ should be redefined not \@glsplural	46	\@GLSuseriv@: set regular format	51
\glxtrtitlelongpl: bug fix: changed \glxtrlong to \glxtrlongpl ..	206	\@GLSuseriv@: set regular format	51
\glxtrtitleshortpl: bug fix: changed \glxtrshort to \glxtrshortpl	202	\@GLSuservi@: set regular format	51
1.04 (2015-04-30)		\@gls@preglossaryhook: added check for entry's existence	139
short-em-footnote: renamed from "footnote-em"	189	\@glsdesc@: set abbreviation and regular format	48
1.04 (2016-05-02)		\@glsdescplural@: set abbreviation and regular format	48
\@glxtrpostloctag: new	39	\@glsfirst@: set abbreviation and regular format	45
\@GLSdesc@: set abbreviation and regular format	48	\@glsfirstplural@: set abbreviation and regular format	47
\@GLSdescplural@: set abbreviation and regular format	49	\@glsname@: set abbreviation and regular format	47
\@GLSfirst@: set abbreviation format ..	46	\@glsplural@: set abbreviation and regular format	46
\@GLSfirstplural@: set abbreviation and regular format	47	\@glsymbol@: set regular format	49
\@GLSname@: set abbreviation and regular format	48		

\@glssymbolplural@: set regular format	49	short-em-nolong-desc: new	187
\@glstext@: set abbreviation and regular format	45	short-em-postfootnote: renamed from “postfootnote-em”	189
\@glstr@deprecated@abbrstyle: new	162	short-footnote: new	169
\@glstr@do@style: new	15	short-long-user: new	196
\@glstr@doloctag: new	39	short-long-user-desc: new	197
\@glstr@idx@entrynumberlist: switched from \let to \newcommand	95	short-nolong: new	173
\@glstr@pagetag: new	38	short-nolong-desc: new	174
\@glstr@pagetag: new	38	short-postfootnote: new	171
\@glstr@preloctag: new	39	short-sc-footnote: renamed from “footnote-sc”	179
\@glstr@postloctag: new	39	short-sc-nolong: new	178
\@glstr@preloctag: new	38	short-sc-nolong-desc: new	178
\glossentrydesc: added glossdescfont attribute check	127	short-sc-postfootnote: renamed from “postfootnote-sc”	180
\Glossentryname: added glossnamefont attribute check	131	short-sm-footnote: renamed from “footnote-sm”	183
\glossentryname: added glossnamefont attribute check	128	short-sm-nolong: new	182
moved post name hook inside condition	130	short-sm-nolong-desc: new	182
\glsabbrvmfont: new	184	short-sm-postfootnote: renamed from “postfootnote-sm”	183
\glsabbrvuserfont: new	190	\letabbreviationstyle: new	162
\glsfirstabbrvmfont: new	184	\newabbreviationstyle: bug fix: corrected test for existence	161
\glsfirstabbrvuserfont: new	190	long-em-noshort-em: new	188
\glsfirstlongemfont: new	184	long-em-noshort-em-desc: new	189
\glsfirstlonguserfont: new	191	long-em-short-em: new	184
\glsifnotregularcategory: new	123	long-em-short-em-desc: new	185
\glslongdefaultfont: new	148	long-noshort: new	176
\glslongemfont: new	184	long-noshort-desc: new	176
\glslongfont: new	148	long-noshort-em: renamed from “long-em”	188
\glslonguserfont: new	190	long-noshort-em-desc: renamed from “long-desc-em”	188
\glstrassignfieldfont: new	44	long-noshort-sc: renamed from “long-sc”	178
\GlsXtrEnablePreLocationTag: new	38	long-noshort-sc-desc: renamed from “long-desc-sc”	179
\glstrfirstscfont: new	176	long-noshort-sm: renamed from “long-sm”	182
\glstrfirstsmfont: new	180	long-noshort-sm-desc: renamed from \long-desc-sm	182
\glstrlongshortdescsort: new	165	long-short-user: new	191
\glstrpostnamehook: added category check	132	long-short-user-desc: new	196
\glstrregularfont: new	40	\renewabbreviationstyle: new style: new	15
\glstruserfield: new	190		
\glstruserparen: new	190	1.05 (2016-06-10)	
\glstrusersuffix: new	191	\eglssetwidest: new	224
\GlsXtrWarnDeprecatedAbbrStyle: new	163	\glsFindWidestAnyName: new	226
short-em-long-em: new	186		
short-em-long-em-desc: new	187		
short-em-nolong: new	187		

\glsFindWidestAnyNameLocation: new	231	\@GLSfirst@: added check for nohyperfirst attribute	46
\glsFindWidestAnyNameSymbol: new	229	\@GLSfirstplural@: added check for nohyperfirst attribute	47
\glsFindWidestAnyNameSymbolLocation: new	230	\@GLSxtrp: new	68
\glsFindWidestLevelTwo: new	227	\@Glsfirst@: added check for nohyperfirst attribute	46
\glsFindWidestUsedAnyName: new	225	\@Glsfirstplural@: added check for nohyperfirst attribute	47
\glsFindWidestUsedAnyNameLocation: new	231	\@Glsxtrp: new	68
\glsFindWidestUsedAnyNameSymbol: new	228	\@gls@preglossaryhook: added \glossxtrsetpopts	139
\glsFindWidestUsedAnyNameSymbolLocation: new	229	\@glsfirst@: added check for nohyperfirst attribute	45
\glsFindWidestUsedLevelTwo: new	226	\@glsfirstplural@: added check for nohyperfirst attribute	47
\glsFindWidestUsedTopLevelName: new	225	\@glsxtrinmark: new	199
\glsfirstlongfootnotefont: new	167	\@glsxtrnotinmark: new	199
\glsgetwidestname: new	224	\@glsxtrp: new	68
\glsgetwidestsubname: new	225	\@glsxtrp@opt: new	67
\glslongfootnotefont: new	167	\glossxtrsetpopts: new	67
\glsxtrAltTreeIndent: new	224	\glsps: new	70
\glsxtralttreeInit: new	224	\glspt: new	70
\glsxtrAltTreePar: new	224	\glsxtr@entry@p: new	69
\glsxtrAltTreeSetHangIndent: new	232	\glsxtrabbrvfootnote: new	168
\glsxtrAltTreeSetSubHangIndent: new	232	\glsxtrchecknohyperfirst: new	45
\glsxtralttreeSubSymbolDescLocation: new	224	\glsxtrfieldtitlecasecs: new	126
\glsxtralttreeSymbolDescLocation: new	223	\glsxtrifinmark: new	199
\glsxtrComputeTreeIndent: new	232	\GLSxtrp: new	71
\glsxtrComputeTreeSubIndent: new	232	\Glsxtrp: new	70
\glsxtrtreetopindent: new	224	\glsxtrp: new	69
short-em-long: fixed incorrect font used by long form	185	\glsxtrsetpopts: new	67
\xglsssetwidest: new	224	short-long-desc: added text key	167
1.06 (2016-06-18)		fixed misspelling of \glsabbrvfont in plural key	167
\@glsdoifexistssorwarn: new	11	long-short-desc: added missing text key	165
\@glsxtr@docdefval: new	10	fixed misspelling of \glsabbrvfont	165
\@glsxtr@useseesee: new	30	footnote: changed first forms to use \glsfirstlongfootnotefont	168
General: disabled docdef key at the start of the document	17	postfootnote: removed \footnote from first keys	169
docdef option changed to choice	10	switched from \glsfirstlongfont to \glsfirstlongfootnotefont	171
\glsxtr@useseesee: new	29	\RestoreAcronyms: modified \@gls@link@checkfirsthyper to set \glsxtrifwasfirstuse	88
\glsxtruseseesee: new	29		
\glsxtruseseeseeformat: new	30		
\if@glsxtrdocdefrestricted: new	11		
1.07 (2016-08-15)		1.08 (2016-12-13)	
\@@glsxtrp: new	67	\@@glsxtr@record: new	6

\@GLS@: added \@glxtr@record	41	\@glxtr@redef@forglsentries: new	5
\@GLSpl@: added \@glxtr@record	41	\@glxtr@shortcutsval: new	13
\@Gls@: added \@glxtr@record	41	\@glxtr@unsrt@getgrouptitle: new	108
\@GLspl@: added \@glxtr@record	41	\@print@noidx@glossary: added	
\@Gls@: added \@glxtr@record	40	redefinition	96
\@Gls@@link@: added		\@glxtr@addloclistfield: added	
\@glxtr@record	42	group key	8
\@Gls@field@link: added		added location key	8
\@glxtr@record	40	\@glxtr@fields: new	104
\@Gls@saveentrycounter: new	17	\@glxtr@linkprefix: new	105
\@Glsdisp: added \@glxtr@record	41	\@glxtr@org@newignoredglossary:	
\@GLspl@: added \@glxtr@record	41	new	25
\@glxtr@dorecord: new	7	\@glxtr@s@newignoredglossary: new	25
\@glxtr@err@undefaction: new	5	\@glxtr@shortcutsval: new	105
\@glxtr@record: new	6	\@glxtr@texencoding: new	104
\@glxtr@warn@onexistsordo: new	5	\@glxtr@writefields: new	105
\@glxtr@warn@undefaction: new	5	\@GlsXtrLoadResources: new	104
\@print@unsrt@glossary: new	107	\@glxtr@resourcefile: changed	
General: added record package option	9	extension to .glstex	104
\@gladd: added \@glxtr@record	44	\@newignoredglossary: added starred	
\@glsoifexists: now defines		version	24
\@glslabel	28	1.12 (2017-02-03)	
\@glxtr@@do@wrglossary: new	17	\@@@glxtr@recordcounter: new	7
\@glxtr@addloclistfield: new	8	\@Gls@preglossaryhook: check for	
\@glxtr@indexonly@saveentrycounter:		definition	139
new	8	\@glxtr@counterrecordhook: new	106
\@glxtr@record: new	106	\@glxtr@display@loc: new	97
\@glxtr@resource: new	104	\@glxtr@docounterrecord: new	106
\@glxtr@saveentrycounter: new	17	\@glxtr@longnewglossaryentry:	
\@glxtr@setup@record: new	8	new	24
\@glxtr@assignfieldfont: added check		\@glxtr@noop@recordcounter: new	7
for existence	44	\@glxtr@op@recordcounter: new	7
\@glxtr@resourcefile: new	104	\@glxtr@provide@storagekey: new	18
\@printunsrt@glossaries: new	107	\@glxtr@s@longnewglossaryentry:	
\@printunsrt@glossary: new	106	new	24
1.09 (2016-12-16)		\@glxtr@entryfmt: new	19
\@glxtr@gettype: new	93	\@glxtr@indexaliased: new	60
\@glxtr@mixed@assign@sortkey:		\@glxtr@setaliasnoindex: new	59
new	93	\@newglossaryentryposthook: added	
\@print@glossary: redefined to save		check for alias key	23
options	93	\@no@glxtr@indexaliased: new	60
\@glxtr@makeglossaries: new	93	\@printunsrt@glossary: new	107
1.10 (2016-12-17)		General: added target key to printgloss	
\@GLSpl@: fixed bug caused by typo in		family	93
command name	41	\@aptoglossarypreamble: new	22
1.11 (2017-01-19)		\@csGlsXtrLetField: new	21
\@glxtr@do@redef@forglsentries:		\@eGlsXtrSetField: new	22
new	5	\@gGlsXtrSetField: new	22
\@glxtr@noidx@do: new	108		

\glsdohyperlink: added check for alias field	63	\glsxtrresourcefile: added catcode change for @	104
\glsnoidxdisplayloc: added redefinition	97	\glsxtrsetaliasnoindex: new	59
\glssettoctitle: added patch	26	\GlsXtrSetField: new	21
\glsxtr@counterrecord: new	106	\glsxtrsetfieldifexists: new	21
\glsxtr@langtag: new	104	\glsxtrunsrtdo: new	108
\glsxtr@newabbreviation: new	145	\Glsxtrusefield: new	21
\glsxtr@org@newignoredglossary: Added check for existence	25	\glsxtrusefield: new	21
\glsxtr@pluralsuffixes: new	105	short-postlong-user: new	194
\glsxtr@provideignoredglossary: new	26	short-postlong-user-desc: new ...	195
\glsxtr@s@newignoredglossary: Added check for existence	25	\longnewglossaryentry: added starred version	23
\glsxtr@s@provideignoredglossary: new	27	long-postshort-user: new	192
\glsxtrabbrvpluralsuffix: new ...	148	long-postshort-user-desc: new ...	193
\glsxtralias: new	23	postdot: new	11
\glsxtrcopytoglossary: new	27	\pretoglossarypreamble: new	23
\glsxtrdeffield: new	21	\print@noop@unsrtglossaryunit: new	108
\glsxtrdisplayendloc: new	98	\print@op@unsrtglossaryunit: new	108
\glsxtrdisplayendlochook: new	98	\printunsrtglossary: added starred form	106
\glsxtrdisplaysingleloc: new	97	\printunsrtglossaryhandler: new .	107
\glsxtrdisplaystartloc: new	98	\printunsrtglossaryunit: new	8
\glsxtrdeffield: new	21	\printunsrtglossaryunitsetup: new	108
\glsxtrentryfmt: new	19	\provideignoredglossary: new	26
\glsxtrfielddolistloop: new	20	\s@glsxtr@provide@storagekey: new	18
\glsxtrfieldforlistloop: new	20	\s@printunsrtglossary: new	107
\glsxtrfieldifinlist: new	20	\xGlsXtrSetField: new	22
\glsxtrfieldlistadd: new	20	1.13 (2017-02-07)	
\glsxtrfieldliststeadd: new	20	\@glsdisp: removed	
\glsxtrfieldlistgadd: new	20	\@glsxtr@org@glsdisp	41
\glsxtrfieldlistxadd: new	20	\glsxtrsetaliasnoindex: switched to \providecommand	59
\glsxtrfieldxifinlist: new	21	1.14 (2017-04-18)	
\glsxtrfmt: new	19	\@gls@link: added redefinition	43
\GlsXtrFmtDefaultOptions: new	19	\@gls@noidx@getgrouptitle: new ...	95
\GlsXtrFmtField: new	19	\@gls@removespaces: new	98
\glsxtrifkeydefined: new	18	\@glsxtr@do@automake@err: new ...	106
\glsxtrindexaliased: new	60	\@glsxtr@org@gloautosee: new	16
\GlsXtrLetField: new	21	\@glsxtr@record: added third arg	6
\GlsXtrLetFieldToField: new	21	\@glsxtr@recordsee: new	7
\GlsXtrLoadResources: removed restriction on only one per document	104	General: added \glsadd option theHvalue	44
\glsxtrlocrangefmt: new	98	added \glsadd option thevalue	43
\glsxtrpostlongdescription: new ..	24	\glsdisablehyper: added redefinition .	63
\glsxtrprovidestoragekey: new	18	\glsenableentrycount: fixed assignment of \@cGls@	75
\GlsXtrRecordCounter: new	106	\glsenableentryunitcount: fixed assignment of \@cGls@	83
\glsxtrresourcecount: new	104		

\glsnavigation:new	96	\glsxtrgetgrouptitle:new	96
\glsxtr@org@getgrouptitle:new ...	95	\glsxtrinitwrgloss:new	42
\glsxtr@recordsee:new	6	\glsxtrlocationhyperlink:new	99
\glsxtr@writefields: added check for automake	105	\glsxtrsetgrouptitle:new	96
\glsxtrdisplayendloc: added check for empty format	98	\glsxtrsupphypernumber:new	99
		\ifglsxtrwrglossbefore:new	42

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	
\.	140, 141, 223
\@	104
\@cGLS@	75, 83
\@cGLSpl@	75, 83
\@cGls@	75, 83
\@cGlspl@	75, 83
\@cgls@	75, 83
\@cglspl@	75, 76, 83
\@do@wrglossary	7, 89
\@do@wrglossary	9, 10, 17, 44, 60
\@glo@assign@sortkey	93
\@glo@list	5
\@glo@type	107
\@gls@expand@field	18
\@glslocalreset	73
\@glslocalunset	73
\@glsreset	73
\@glsunset	72
\@glsxtr@autoindex@escspch	135, 136
\@glsxtr@checkspch	133, 134, 136
\@glsxtr@disabledflycommand	35
\@glsxtr@record	9, 10
\@glsxtr@recordcounter	9, 10, 106
\@glsxtrp	68
\@glsxtrpostloctag	38
\@glsxtrpreloctag	38
\@newglossaryentry@defcounters	74
\@newglossaryentry@defunitcounters	82
\@par	224
\@ACRlong	64
\@ACRlongpl	64
\@ACRshort	64
\@ACRshortpl	64
\@Acrlong	64
\@Acrlongpl	64
\@Acrshort	64
\@Acrshortpl	64
\@GLS@	64, 77, 78
\@GLSdesc@	49
\@GLSpl@	64, 78, 79
\@GLSplural@	65
\@GLSsymbol@	50
\@GLStext@	64
\@GLSxtr@full	150
\@GLSxtr@fullpl	151
\@GLSxtr@p@acrlong@	64
\@GLSxtr@p@acrlongpl@	64
\@GLSxtr@p@acrshort@	64
\@GLSxtr@p@acrshortpl@	64
\@GLSxtr@p@long@	64
\@GLSxtr@p@longpl@	64
\@GLSxtr@p@plural@	64
\@GLSxtr@p@short@	64
\@GLSxtr@p@shortpl@	64
\@GLSxtr@p@text@	64
\@GLSxtrlong	64, 154, 155
\@GLSxtrlongpl	64, 158
\@GLSxtrp	71, 72
\@GLSxtrshort	64, 153
\@GLSxtrshortpl	64, 156
\@Gls@	64, 77, 78
\@Gls@acentryname	86
\@Gls@entry@field	56, 70, 71
\@Gls@entryname	86
\@GlsXtrEnableOnTheFly	32
\@Glspl@	64, 77, 78
\@Glsplural@	65
\@Glstext@	64
\@Glsxtr	33, 35
\@Glsxtr@full	149
\@Glsxtr@fullpl	151
\@Glsxtr@p@acrlong@	64
\@Glsxtr@p@acrlongpl@	64

\@Glsxtr@p@acrshort@	64	\@glo@counterprefix	7, 98
\@Glsxtr@p@acrshortpl@	64	\@glo@countunit	79, 80
\@Glsxtr@p@long@	64	\@glo@default@sorttype	91
\@Glsxtr@p@longpl@	64	\@glo@desc	24
\@Glsxtr@p@plural@	64	\@glo@descplural	24
\@Glsxtr@p@short@	64	\@glo@group	9
\@Glsxtr@p@shortpl@	64	\@glo@label	8, 9, 18, 23, 29, 30, 56, 63, 225–232
\@Glsxtr@p@text@	64	\@glo@location	8
\@Glsxtrlong	64, 154	\@glo@loclist	8
\@Glsxtrlongpl	64, 157	\@glo@name	133
\@Glsxtrp	70, 71	\@glo@no@assign@sortkey	93
\@Glsxtrpl	34, 35	\@glo@parent	226–228
\@Glsxtrshort	64, 152	\@glo@see	23, 29, 30
\@Glsxtrshortpl	64, 156	\@glo@sort	133
\@acrlong	64	\@glo@sorttype	91, 96, 97
\@acrlongpl	64	\@glo@text	42
\@acrshort	64	\@glo@thislettergrp	109
\@acrshortpl	64	\@glo@thisvalue	190
\@alt@glshyp@opt	61	\@glo@tmp	18, 56
\@auxout	7, 8, 39, 75, 84, 88, 89, 99, 100, 104–106	\@glo@type	30, 86, 89, 92, 93, 96, 97, 99, 100, 102, 103, 107
\@bibgls@restoreat	104	\@glo@types	124, 225–231
\@cGLS	78	\@glossary@default@style	35, 36, 92, 234
\@cGLS@	75, 78, 83	\@glossarystyle	92
\@cGLSpl	78	\@gls@	64, 76, 78
\@cGLSpl@	75, 78, 83	\@gls@@link	42
\@cGls@	75, 83	\@gls@ReturnAfterFi	98
\@cGlspl@	75, 83	\@gls@actualchar	134
\@cgls@	75, 83	\@gls@adjustmode	44
\@cglspl@	75, 83	\@gls@alt@hyp@opt	62
\@disable@onlypremakeg	89	\@gls@alt@hyp@opt@char	61, 62
\@do@auxoutstuff	99, 100	\@gls@alt@hyp@opt@keys	62
\@do@glsee	23	\@gls@automake	91
\@do@newglossaryentry	86, 146	\@gls@between	96
\@do@seeglossary	9, 10, 16, 89	\@gls@checkedmkidx	133, 134, 136
\@do@wrglossary	43	\@gls@checkmkidxchars	133
\@empty	44, 52–56, 133, 134, 149–158	\@gls@codepage	100
\@end@glxtr@addunused	30	\@gls@counter	6, 7, 43, 44, 60
\@end@glxtr@gettype	91, 93	\@gls@currentlettergroup	97, 107, 109
\@end@glxtr@usesee	29, 30	\@gls@declareoption	4
\@endfortrue	161	\@gls@doautomake	91, 106
\@firstofone	44, 45, 127, 128, 132, 138	\@gls@doautomake@err	106
\@firstofthree	41, 44, 52–55, 61, 62, 149, 150, 152, 154, 155, 157	\@gls@encapchar	134
\@firstoftwo	45–50, 53–56, 58, 61, 88, 142, 143, 149–151, 155–158, 199, 200	\@gls@entry@count	75
\@for	5, 15, 31, 74, 85, 89, 91, 96, 107, 126, 137	\@gls@entry@field	18, 21, 56, 69–72, 74
\@glo@assign@sortkey	91	\@gls@entry@unitcount	83, 84
\@glo@autosee	9, 10, 16	\@gls@field@font	44–52
\@glo@category	79	\@gls@field@link	45–52, 57, 58
		\@gls@getgrouptitle	96, 107

<code>\@gls@grptitle</code>	96	<code>\@glsacronymlists</code>	86
<code>\@gls@hyp@opt</code>	57, 58, 62, 78, 149–158	<code>\@glsdoifexistsorwarn</code> ...	10, 128, 129, 131
<code>\@gls@hyp@opt@cs</code>	61, 62	<code>\@glsentry</code>	75, 84
<code>\@gls@increment@currcount</code>	74	<code>\@glslink</code>	43, 63, 64
<code>\@gls@increment@currunitcount</code>	82	<code>\@glslocref</code>	7
<code>\@gls@keymap</code>	8, 9, 18, 29, 56, 105	<code>\@glsnextpages</code>	92
<code>\@gls@label</code>	7, 61, 89, 106, 161	<code>\@glsnonextpages</code>	92
<code>\@gls@levelchar</code>	134	<code>\@glsnumberformat</code> ..	6, 7, 43, 44, 60, 132, 133
<code>\@gls@link</code>	19, 40, 42, 52–56, 149–158	<code>\@glsorder</code>	88, 89
<code>\@gls@link@checkfirsthyper</code>	41, 88	<code>\@glspl@</code>	64, 77, 78
<code>\@gls@link@label</code>	43	<code>\@glsplural@</code>	65
<code>\@gls@link@nocheckfirsthyper</code>	40, 52–56, 149–158	<code>\@glsplunc@token</code>	142
<code>\@gls@link@opts</code>	43	<code>\@glsstyle@alttree</code>	223
<code>\@gls@list</code>	96	<code>\@glsstyle@inline</code>	223
<code>\@gls@local@increment@currcount</code>	74	<code>\@glsstyle@listdotted</code>	218
<code>\@gls@local@increment@currunitcount</code> ..	83	<code>\@gls@target</code>	63, 93
<code>\@gls@location</code>	109, 110	<code>\@gls@text@</code>	64
<code>\@gls@loclist</code>	94, 95, 109, 110	<code>\@glswidestname</code>	224, 225, 232
<code>\@gls@longpl</code>	144–146	<code>\@glsxtr</code>	33, 35
<code>\@gls@nohyperlist</code>	25, 27	<code>\@glsxtr@do@wrglossary</code>	89
<code>\@gls@noidx@do</code>	97	<code>\@glsxtr@abbreviationsdef</code>	12, 16
<code>\@gls@noidx@nosanitizesort</code>	91	<code>\@glsxtr@activate@initialtagging</code> ..	138, 139
<code>\@gls@noidx@sanitizesort</code>	91	<code>\@glsxtr@addunitcounter</code>	80
<code>\@gls@noidx@loclist@finalsep</code>	94	<code>\@glsxtr@addunusedxrefs</code>	30, 31
<code>\@gls@noidx@loclist@prev</code>	94	<code>\@glsxtr@attrval</code> ...	127, 128, 130, 131, 133
<code>\@gls@noidx@loclist@sep</code>	94	<code>\@glsxtr@autoindex@at</code>	133–135
<code>\@gls@noref@warn</code>	90, 97	<code>\@glsxtr@autoindex@doextra@esc</code>	133
<code>\@gls@org@glsnoidx@displayloc</code>	94	<code>\@glsxtr@autoindex@encap</code>	133–135
<code>\@gls@org@glssee@format</code>	94, 95	<code>\@glsxtr@autoindex@esc</code>	133–136
<code>\@gls@preglossaryhook</code>	93, 138	<code>\@glsxtr@autoindex@escat</code>	134, 135
<code>\@gls@prevlevel</code>	233	<code>\@glsxtr@autoindex@escencap</code> ...	134, 135
<code>\@gls@quotechar</code>	133	<code>\@glsxtr@autoindex@esclevel</code> ...	134, 135
<code>\@gls@reference</code>	31, 89	<code>\@glsxtr@autoindex@escquote</code> ...	133, 135
<code>\@gls@saveentrycounter</code> ...	9, 10, 17, 43, 44	<code>\@glsxtr@autoindex@level</code>	134, 135
<code>\@gls@see@noindex</code>	104	<code>\@glsxtr@autoindex@setname</code>	133
<code>\@gls@setdefault@glslink@opts</code> ...	43, 60	<code>\@glsxtr@autoindex@crossrefs</code> ...	9, 11, 29
<code>\@gls@setsort</code>	43	<code>\@glsxtr@cat</code>	74, 85, 137
<code>\@gls@short</code>	145	<code>\@glsxtr@counterrecordhook</code>	7
<code>\@gls@shortpl</code>	144–146	<code>\@glsxtr@csname</code>	80, 81, 83
<code>\@gls@sort</code>	109	<code>\@glsxtr@current@style</code>	36, 234
<code>\@gls@tmp</code>	96	<code>\@glsxtr@currentunitcount</code>	80, 81, 83
<code>\@gls@tmpb</code>	136	<code>\@glsxtr@currunitcount</code>	82, 84
<code>\@gls@type</code>	89–91, 161, 225–231	<code>\@glsxtr@declareoption</code>	4, 11, 12, 14
<code>\@gls@write@entrycounts</code>	75	<code>\@glsxtr@defaultnoglossarywarning</code> ..	14, 15
<code>\@gls@write@entryunitcounts</code>	83	<code>\@glsxtr@deprecated@abbrstyle</code>	179, 180, 182, 183, 188–190
<code>\@gls@write@entryunitcounts@do</code>	84	<code>\@glsxtr@disabledflycommand</code>	35
<code>\@gls@xref</code>	8	<code>\@glsxtr@display@loc</code>	97
<code>\@gls@abbrv@current@abbreviation</code> ..	145, 158		

\@glxtr@do@wrindex	61	\@glxtr@noidx@do	108
\@glxtr@do@glstdisablehyperinlist	58, 59	\@glxtr@noidx@entrynumberlist	90
\@glxtr@do@redef@forglsentries	6	\@glxtr@noidx@getgrouptitle	107
\@glxtr@do@style	15, 216	\@glxtr@noidx@numberlistloop	90
\@glxtr@do@titlecaps@warn	127–130, 138	\@glxtr@noop@recordcounter	7, 9
\@glxtr@do@abbreviationsdef	12	\@glxtr@notfoundinlist	143
\@glxtr@do@accsupp	14, 16	\@glxtr@op@recordcounter	9, 10
\@glxtr@docdefval	10, 11, 31	\@glxtr@optlist	34, 35
\@glxtr@docounterrecord	7	\@glxtr@org@GLS@	41
\@glxtr@doloctag	38	\@glxtr@org@GLSpl@	41
\@glxtr@dorecord	7	\@glxtr@org@Gls@	41
\@glxtr@dostylewarn	161	\@glxtr@org@Glspl@	41
\@glxtr@enabletagging	137	\@glxtr@org@Glsxtrtitlefirst	199, 200
\@glxtr@end@	32	\@glxtr@org@Glsxtrtitlefirstplural	199, 201
\@glxtr@endescspch	134–136	\@glxtr@org@Glsxtrtitlefull	200, 201
\@glxtr@entrycount@org@localreset	75	\@glxtr@org@Glsxtrtitlefullpl	200, 201
\@glxtr@entrycount@org@localunset	74	\@glxtr@org@Glsxtrtitlelong	200, 201
\@glxtr@entrycount@org@reset	75	\@glxtr@org@Glsxtrtitlelongpl	200, 201
\@glxtr@entrycount@org@unset	74	\@glxtr@org@Glsxtrtitleplural	199, 200
\@glxtr@entryunitcount@org@localreset	83	\@glxtr@org@Glsxtrtitleshort	199, 200
\@glxtr@entryunitcount@org@localunset	82, 83	\@glxtr@org@Glsxtrtitleshortpl	199, 200
\@glxtr@entryunitcount@org@reset	83	\@glxtr@org@Glsxtrtitletext	199, 200
\@glxtr@entryunitcount@org@unset	82	\@glxtr@org@MakeUppercase	199, 200
\@glxtr@err@undefaction	6, 9	\@glxtr@org@checkfirsthyper	58, 88
\@glxtr@field@linkdefs	40	\@glxtr@org@delimN	38, 39
\@glxtr@format@overridefalse	132	\@glxtr@org@delimR	38, 39
\@glxtr@format@overridetrue	132, 133	\@glxtr@org@doseeglossary	9, 10, 89
\@glxtr@foundinlist	143	\@glxtr@org@gloautosee	9, 10
\@glxtr@full	149	\@glxtr@org@gls@	40, 41
\@glxtr@fullpl	150	\@glxtr@org@glsignore	38, 39
\@glxtr@gettype	91	\@glxtr@org@glspl@	41
\@glxtr@glossdescfont	127, 128	\@glxtr@org@glsxtrtitlefirst	199, 200
\@glxtr@glossnamefont	128–132	\@glxtr@org@glsxtrtitlefirstplural	199, 201
\@glxtr@gobbleto@endescspch	136	\@glxtr@org@glsxtrtitlefull	200, 201
\@glxtr@idx@displaynumberlist	90	\@glxtr@org@glsxtrtitlefullpl	200, 201
\@glxtr@idx@entrynumberlist	90	\@glxtr@org@glsxtrtitlelong	199, 201
\@glxtr@ifcsstart	32	\@glxtr@org@glsxtrtitlelongpl	199, 201
\@glxtr@ifpunctoken	143	\@glxtr@org@glsxtrtitleplural	199, 200
\@glxtr@ifunitcounter	79	\@glxtr@org@glsxtrtitleshort	199, 200
\@glxtr@insert@dots	144	\@glxtr@org@glsxtrtitleshortpl	199, 200
\@glxtr@insert@dots@next	144, 145	\@glxtr@org@glsxtrtitletext	199, 200
\@glxtr@insert@dots	145	\@glxtr@org@makeglossaries	88
\@glxtr@label	31, 126	\@glxtr@org@markboth	198, 199
\@glxtr@loadstyles	217	\@glxtr@org@markright	198, 199
\@glxtr@longnewglossaryentry	23	\@glxtr@org@newacronymstyle	87
\@glxtr@mixed@assign@sortkey	91	\@glxtr@org@postdescription	139
\@glxtr@noidx@displaynumberlist	90	\@glxtr@org@see@noindex	104

<code>\@glxstr@org@setacronymstyle</code>	87	<code>\@glxstrinmark</code>	198, 199
<code>\@glxstr@orgprintglossary</code>	35, 93	<code>\@glxstrlong</code>	64, 153
<code>\@glxstr@orgwarndep</code>	144	<code>\@glxstrlongpl</code>	64, 157
<code>\@glxstr@p@acrlong@</code>	64	<code>\@glxstrnotinmark</code>	198, 199
<code>\@glxstr@p@acrlongpl@</code>	64	<code>\@glxstrp</code>	69, 70
<code>\@glxstr@p@acrshort@</code>	64	<code>\@glxstrp@opt</code>	67
<code>\@glxstr@p@acrshortpl@</code>	64	<code>\@glxstrpl</code>	34, 35
<code>\@glxstr@p@long@</code>	64	<code>\@glxstrpostloctag</code>	37–39
<code>\@glxstr@p@longpl@</code>	64	<code>\@glxstrpreloctag</code>	37–39
<code>\@glxstr@p@plural@</code>	64	<code>\@glxstrsetaliasnoindex</code>	59, 60
<code>\@glxstr@p@short@</code>	64	<code>\@glxstrshort</code>	64, 152
<code>\@glxstr@p@shortpl@</code>	64	<code>\@glxstrshortpl</code>	64, 155
<code>\@glxstr@p@text@</code>	64	<code>\@glxstrundeftag</code>	5, 17
<code>\@glxstr@pagestag</code>	38	<code>\@gobble</code>	6, 9, 11, 45, 144
<code>\@glxstr@pagetag</code>	38	<code>\@gobbletwo</code>	144
<code>\@glxstr@prevunitcount</code>	82	<code>\@ifnextchar</code>	61
<code>\@glxstr@printglossopts</code>	35, 91, 93	<code>\@ifpackageloaded</code>	...
<code>\@glxstr@provide@addstoragekey</code>	19		4, 12, 105, 110, 126, 128, 131, 132, 215
<code>\@glxstr@provide@storagekey</code>	18	<code>\@ifstar</code>	18, 23, 24, 26, 32, 61, 106, 137
<code>\@glxstr@record</code>	9, 10, 40–42, 44	<code>\@ifundefined</code>	215
<code>\@glxstr@recordsee</code>	9	<code>\@ignored@glossaries</code>	25–27
<code>\@glxstr@redef@for@lsentries</code>	6, 16, 17	<code>\@input</code>	104
<code>\@glxstr@redef@styles</code>	15, 216	<code>\@input@</code>	99
<code>\@glxstr@reg@glosslist</code>	88–91, 93	<code>\@istfilename</code>	89
<code>\@glxstr@s@longnewglossaryentry</code>	23	<code>\@makeglossary</code>	89
<code>\@glxstr@savepreloctag</code>	38, 39	<code>\@mfu@domakefirstuc</code>	138
<code>\@glxstr@setentrycountunsetattr</code>	73	<code>\@mfu@nocaplist</code>	138
<code>\@glxstr@setentryunitcountunsetattr</code>	85	<code>\@one</code>	75, 84
<code>\@glxstr@setupshortcuts</code>	13, 14, 16, 17	<code>\@newglossaryentry@defcounters</code>	74, 82
<code>\@glxstr@shortcutsval</code>	13, 105	<code>\@newglossaryentryposthook</code>	8, 9, 18, 56
<code>\@glxstr@swaptwo</code>	143	<code>\@newglossaryentryprehook</code>	8, 9, 18, 24, 56
<code>\@glxstr@tag</code>	138	<code>\@nil</code>	98, 109
<code>\@glxstr@taggingcs</code>	138	<code>\@nnil</code>	133, 134, 136, 143, 144
<code>\@glxstr@theHvalue</code>	6, 7, 44	<code>\@no@glxstrindexaliased</code>	59, 60
<code>\@glxstr@thevalue</code>	6, 7, 43, 44	<code>\@no@makeglossaries</code>	103
<code>\@glxstr@thisloctag</code>	38, 39	<code>\@nopostdesc</code>	92
<code>\@glxstr@titlelabel</code>	95, 96, 108	<code>\@onelevel@sanitize</code>	8, 34, 95, 96, 108
<code>\@glxstr@tmp</code>	15, 98	<code>\@onlypreamble</code>	...
<code>\@glxstr@type</code>	126		35, 38, 84, 104, 106, 133, 135, 137
<code>\@glxstr@unitcountlist</code>	80	<code>\@org@glossaryentrynumbers</code>	92, 93
<code>\@glxstr@unsrt@getgrouptitle</code>	107	<code>\@org@newglossaryentryprehook</code>	24
<code>\@glxstr@usesee</code>	30	<code>\@print@unsrt@glossary</code>	107
<code>\@glxstr@warn@onexistsordo</code>	6, 9, 10	<code>\@printgloss@setsort</code>	91, 92
<code>\@glxstr@warn@undefaction</code>	6, 9, 10	<code>\@printglossary</code>	35, 107
<code>\@glxstr@docdeffalse</code>	31	<code>\@printunsrtglossary</code>	106
<code>\@glxstr@entryfmt</code>	19	<code>\@sGlsXtrEnableOnTheFly</code>	32
<code>\@glxstrindexaliased</code>	59	<code>\@secondofthree</code>	45–
<code>\@glxstrindexcrossrefsfalse</code>	11		47, 52–55, 57, 150, 151, 153, 154, 156, 157
<code>\@glxstrindexcrossrefstrue</code>	11		

\@secondoftwo	41, 44, 48–56, 58, 63, 88, 93, 143, 149, 150, 152–158, 170, 199, 200	\AFP	13
\@sglsxtr@provide@storagekey	18	\Afp	12
\@thirdofthree	45–47, 53–56, 58, 150, 151, 153, 155, 156, 158	\afp	12
\@thirdoftwo	48–52	\AL	12
\@warn@nomakeglossaries	100	\Al	12
\@xdy@main@language	99	\al	12
\@xdy@language	99, 100	\ALP	12
\\	98	\Alp	12
		\alp	12
		\AnyTrackedLanguages	215
		\appto	8, 9, 15, 18, 23, 29, 56, 61, 74, 82, 108, 132, 142, 145
_	101, 102	\AS	12
		\As	12
A		\as	12
\AB	12	\ASP	12
\Ab	12	\Asp	12
\ab	12	\asp	12
abbreviation styles:		\AtBeginDocument	17, 36, 37, 105
long-noshort	188	\AtEndDocument	30, 75, 83, 99, 100
long-postshort-user	193		
long-short-user	192	B	
short	173	babel package	133, 134, 142
short-long-user	194	\begin	97, 101, 102, 107, 218–222
short-postlong-user	196	\begingroup	6, 60, 107
\abbreviationsname	12	\bgroup	24, 92
\abbrvpluralsuffix			
....	105, 145, 146, 164, 166, 168, 170, 172, 173, 175, 177–183, 191, 192, 195, 197	C	
\ABP	12	\catcode	104
\Abp	12	category attributes:	
\abp	12	discardperiod	141
\ACRfullfmt	86	entrycount	72–75, 85
\Acrfullfmt	86	firstuc	130
\acrfullfmt	86	glossdesc	126
\ACRfullplfmt	87	glossdescfont	127
\Acrfullplfmt	86	glossname	128
\acrfullplfmt	86	glossnamefont	128, 131
\acronymentry	86	headuc	201
\acronymfont	52–56, 66, 87	indexname	133
\acronymname	12	indexonlyfirst	60
\acronymsort	86	insertdots	145, 146
\acronymtype	12, 86, 87	nohyper	58
\acrpluralsuffix	86, 105	nohyperfirst	45–47
\actualchar	136	regular	39, 79, 163–168, 170, 172, 174, 175, 185, 186, 191, 197
\addtolength	233	\cGLS	12, 73, 85
\advance	75, 84, 104	\cGls	12, 73, 85
\AF	13	\cglS	12, 73, 85
\Af	12	\cGLSformat	77
\af	12	\cGlsformat	77

<code>\Glsaccessfirstplural</code>	47	<code>\glsdefaulttype</code>	
<code>\glsaccessfirstplural</code>	47	5, 11, 22, 23, 91, 92, 101, 107, 108
<code>\Glsaccesslong</code>		<code>\glsdescriptionaccessdisplay</code>	114, 115, 127
.....	54, 147, 154, 165, 172, 175, 192, 193	<code>\glsdescriptionpluralaccessdisplay</code>	115
<code>\glsaccesslong</code>	54, 55, 147, 154, 155, 164,	<code>\glsdescwidth</code>	218, 220–222
	166, 168, 169, 171–175, 191, 193, 195, 197	<code>\glsdetoklabel</code>	20–22,
<code>\Glsaccesslongpl</code>			24, 28–32, 43, 44, 60, 63, 74, 75, 80–84,
.....	55, 147, 157, 165, 172, 175, 192, 193		89, 92, 94, 95, 109, 126, 129, 130, 226–228
<code>\glsaccesslongpl</code>	55, 56, 147, 157, 158, 164,	<code>\glsdisplaynumberlist</code>	90, 93
	166, 167, 169, 171–175, 191, 193, 195, 197	<code>\glsdohyperlink</code>	64
<code>\GLSaccessname</code>	48	<code>\glsdohypertarget</code>	93
<code>\Glsaccessname</code>	48	<code>\glsdoifexists</code>	10,
<code>\glsaccessname</code>	47		21, 27, 29, 40, 41, 44, 52–56, 94, 95, 149–158
<code>\GLSaccessplural</code>	47	<code>\glsdoifexistsordo</code>	19, 42
<code>\Glsaccessplural</code>	46	<code>\glsdoifexistsorwarn</code>	11, 126, 127, 136, 137
<code>\glsaccessplural</code>	46	<code>\glsdoifnoexists</code>	24
<code>\Glsaccessshort</code>		<code>\glsdonohyperlink</code>	43, 63
.	52, 153, 159, 166, 169, 171, 174, 195, 197	<code>\glsdosanitizesort</code>	91
<code>\glsaccessshort</code> ...	52, 53, 147, 152, 153,	<code>\glsenableentrycount</code>	73, 76, 84
	159, 164–166, 168–175, 191–193, 195, 197	<code>\glsenableentryunitcount</code>	75, 85
<code>\Glsaccessshortpl</code>		<code>\glsentrycounter</code>	98
.	53, 156, 159, 167, 169, 171, 174, 195, 197	<code>\glsentrycurrcount</code>	74, 75, 82
<code>\glsaccessshortpl</code>	53, 54, 147, 155, 156,	<code>\Glsentrydesc</code>	114, 119, 128
	159, 164–166, 169–175, 191–193, 195, 197	<code>\glsentrydesc</code>	114, 115, 119, 128
<code>\GLSaccesssymbol</code>	49	<code>\Glsentrydescplural</code>	115, 119
<code>\Glsaccesssymbol</code>	49, 137	<code>\glsentrydescplural</code>	115, 119
<code>\glsaccesssymbol</code>	49, 137, 141	<code>\Glsentryfirst</code>	79, 112, 118
<code>\GLSaccesssymbolplural</code>	50	<code>\glsentryfirst</code>	79, 112, 118, 212
<code>\Glsaccesssymbolplural</code>	50	<code>\Glsentryfirstplural</code>	79, 113, 118
<code>\glsaccesssymbolplural</code>	49	<code>\glsentryfirstplural</code>	79, 112, 113, 118, 212
<code>\GLSaccessstext</code>	45	<code>\glsentryfmt</code>	25–27
<code>\Glsaccessstext</code>	45	<code>\Glsentryfull</code>	87
<code>\glsaccessstext</code>	45	<code>\glsentryfull</code>	87
<code>\glsacrshortcutstrue</code>	13, 14	<code>\Glsentryfullpl</code>	87
<code>\glsacspacemax</code>	88	<code>\glsentryfullpl</code>	87
<code>\glsadd</code>	31, 101	<code>\glsentryitem</code>	218–222, 233
<code>\glsaddstoragekey</code>	23, 121	<code>\Glsentrylong</code>	66, 67, 79, 117, 120
<code>\glsbackslash</code>	32	<code>\glsentrylong</code>	66,
<code>\glscapscase</code>	41, 44–58, 149–160		67, 79, 116, 117, 120, 170, 194, 196, 213
<code>\glscategory</code>		<code>\Glsentrylongpl</code>	66, 67, 79, 117, 120
.....	40, 44, 58, 65, 66, 122, 123, 126–	<code>\glsentrylongpl</code>	66, 67, 79, 117, 120, 213, 214
	128, 130–132, 137, 140, 149–153, 155, 156	<code>\Glsentryname</code>	110, 117, 129–132
<code>\glscategorylabel</code>		<code>\glsentryname</code>	110, 111, 117, 118, 133, 225–232
.....	58, 144–146, 170, 192, 194, 196	<code>\glsentrynumberlist</code>	90, 95, 230–232
<code>\glsclosebrace</code>	102, 103	<code>\Glsentryplural</code>	111, 118
<code>\glscurrententrylabel</code>		<code>\glsentryplural</code>	111, 112, 118, 211
.....	37–39, 92, 99, 107, 132, 139, 140	<code>\glsentryprevcount</code>	74, 76, 82
<code>\glscurrentfieldvalue</code>	19, 20, 190	<code>\glsentryprevmaxcount</code>	82
<code>\glscustomtext</code> ..	40, 41, 52–56, 149–158, 160	<code>\glsentryprevtotalcount</code>	82

<code>\Glsentryshort</code>	65, 66, 116, 120	<code>\glsgetattribute</code>	
<code>\glsentryshort</code>	65,		62, 76, 80–82, 99, 127, 128, 130, 131, 133
	66, 88, 115, 116, 119, 120, 192, 194, 209, 210	<code>\glsgetcategoryattribute</code>	122
<code>\Glsentryshortpl</code>	65, 66, 116, 120	<code>\glsgetwidestname</code>	224
<code>\glsentryshortpl</code>	65, 66, 116, 120, 210	<code>\glsgroupheading</code>	109, 218–222, 233
<code>\Glsentrysymbol</code>	113, 119	<code>\glsgroupskip</code>	109, 219–223, 234
<code>\glsentrysymbol</code>	113, 119, 229, 230	<code>\glshasattribute</code>	62, 75, 76, 80, 81,
<code>\Glsentrysymbolplural</code>	114, 119		83, 84, 99, 127, 128, 130, 131, 133, 164–
<code>\glsentrysymbolplural</code>	114, 119		168, 170, 185, 186, 191, 192, 194, 196, 197
<code>\Glsentrytext</code>	111, 118	<code>\glshascategoryattribute</code>	122
<code>\glsentrytext</code>	63, 111, 118, 210, 211	<code>\glshyperlink</code>	63
<code>\Glsentryuseri</code>	50	<code>\glshypernavsep</code>	96
<code>\glsentryuseri</code>	50	<code>\glshypernumber</code>	99, 132
<code>\Glsentryuserii</code>	50	<code>\glsifattribute</code>	42, 45, 59, 60,
<code>\glsentryuserii</code>	50		69, 124, 127–130, 139, 141, 142, 201–209
<code>\Glsentryuseriii</code>	51	<code>\glsifcategory</code>	124
<code>\glsentryuseriii</code>	51	<code>\glsifcategoryattribute</code>	
<code>\Glsentryuseriv</code>	51		58, 122, 123, 145, 146
<code>\glsentryuseriv</code>	51	<code>\glsifnotregular</code>	44
<code>\Glsentryuserv</code>	51	<code>\glsifnotregularcategory</code>	123
<code>\glsentryuserv</code>	51	<code>\glsifplural</code>	
<code>\Glsentryuservi</code>	52		41, 44, 46–50, 52–56, 141, 142, 149–159
<code>\glsentryuservi</code>	52	<code>\glsifregular</code>	40, 44, 79
<code>\glsfieldfetch</code>	63	<code>\glsifregularcategory</code>	123
<code>\glsfieldxdef</code>	126	<code>\glsifusetranslator</code>	26
<code>\glsfindwidesttoplevelname</code>	225	<code>\glsignore</code>	38, 39
<code>\GLSfirst</code>	204, 205	<code>\glsinlinedescformat</code>	223
<code>\Glsfirst</code>	205	<code>\glsinlinesubdescformat</code>	223
<code>\glsfirst</code>	204	<code>\glsinsert</code>	42, 44, 52–56, 149–160
<code>\glsfirstabbrvdefaultfont</code>		<code>\glskeylisttok</code>	86, 145, 146
	148, 164, 166, 168, 170, 172, 173, 175	<code>\glslabel</code>	28, 40, 42, 43, 58–60, 62,
<code>\glsfirstabbrvmfont</code>	185–190		63, 88, 140, 141, 158–160, 170, 192, 194, 196
<code>\glsfirstabbrvfont</code>	87,	<code>\glslabeltok</code>	
	147, 164–175, 177–183, 185–193, 195, 197		86, 145, 146, 164–168, 170, 172–
<code>\glsfirstabbrvuserfont</code> ..	191, 192, 194–197		174, 176, 184–186, 188, 191–194, 196, 197
<code>\glsfirstaccessdisplay</code>	112	<code>\glsletentryfield</code>	133
<code>\glsfirstlongdefaultfont</code>		<code>\glslink</code>	86, 87
	164, 166, 172, 173, 175	<code>\glslink options</code>	
<code>\glsfirstlongemfont</code>	185–189	format	132
<code>\glsfirstlongfont</code>	147,	hyper	198
	164–168, 170, 172–176, 184–189, 191–197	noindex	6, 59, 198
<code>\glsfirstlongfootnotefont</code>	168–171	wrgloss	42
<code>\glsfirstlonguserfont</code> ..	191, 192, 195, 197	<code>\glslinkcheckfirsthyperhook</code>	58
<code>\GLSfirstplural</code>	205	<code>\glslinkpostsetkeys</code>	43
<code>\Glsfirstplural</code>	205, 206	<code>\glslinkvar</code>	61, 62
<code>\glsfirstplural</code>	205	<code>\glslistdottedwidth</code>	218
<code>\glsfirstpluralaccessdisplay</code> ..	112, 113	<code>\glslocalunset</code>	42
<code>\glsforeachincategory</code>	161	<code>\glslongaccessdisplay</code>	116, 117
<code>\glsgenentryfmt</code>	40		

<code>\glslongdefaultfont</code>	<code>\glssetattribute</code> 164–168, 170, 172–174, 176, 185, 186, 188, 191, 192, 194, 196, 197
..... 148, 164, 166, 167, 172, 173, 175	<code>\glssetcategoryattribute</code>
<code>\glslongemfont</code> 74, 85, 88, 122, 123, 125, 137
..... 184–189	<code>\glssetnoexpandfield</code>
<code>\glslongfont</code>	8, 9
..... 66, 148, 154, 155, 157, 158, 164, 166, 168, 170, 172, 173, 175, 185–189, 191, 192, 195, 197	<code>\glssettoctitle</code>
<code>\glslongfootnotefont</code>	92
..... 167, 168, 170	<code>\glsshortaccessdisplay</code>
<code>\glslongpltok</code>	115, 116
..... 146, 164–168, 174, 176, 185, 186, 188, 191–194, 196, 197	<code>\glsshortpltok</code>
<code>\glslongpluralaccessdisplay</code>	146, 164–168, 170–173, 185, 186, 191, 192, 194, 196, 197
..... 117	<code>\glsshortpluralaccessdisplay</code>
<code>\glslongtok</code> 86, 145, 146, 164–168, 170, 172– 174, 176, 184–186, 188, 191–194, 196, 197	116
<code>\glslonguserfont</code> ...	<code>\glsshorttok</code>
191, 192, 194, 195, 197 86, 145, 146, 164–168, 170, 171, 173, 176, 184–186, 188, 191–194, 196, 197
<code>\glsnameaccessdisplay</code>	<code>\glssubentryitem</code>
110, 129, 131	218–223, 233
<code>\glsnamefont</code>	<code>\glssymbolaccessdisplay</code>
128–132	113
<code>\glsnavhyperlink</code>	<code>\glssymbolpluralaccessdisplay</code>
96	114
<code>\glsnextpages</code>	<code>\glstarget</code>
92	218–223, 233
<code>\glsnoidxdisplayloc</code>	<code>\GLStext</code>
94	203
<code>\glsnoidxdisplaylocclsthandler</code>	<code>\Glstext</code>
94	203
<code>\glsnoidxlocclst</code>	<code>\glstext</code>
95, 109, 110	203
<code>\glsnoidxnumberlistloopclsthandler</code>	<code>\glstextaccessdisplay</code>
94	111
<code>\glsnonnextpages</code>	<code>\glstextformat</code>
92	42, 43
<code>\glsnonnumberlistfalse</code>	<code>\glstextup</code>
37	176
<code>\glsnonnumberlisttrue</code>	<code>\glstreeindent</code>
37	232, 233
<code>\glsnumberlistloop</code>	<code>\glstreenamebox</code>
90	233
<code>\glsnumlistlastsep</code>	<code>\glstreenamefmt</code>
94	224–233
<code>\glsnumlistsep</code>	<code>\GLstrLetField</code>
94	21
<code>\glsopenbrace</code>	<code>\glstype</code>
102, 103	42, 43, 52–56, 149–158
<code>\glsorder</code>	<code>\glunset</code>
89	31, 42, 76, 77
<code>\glspagelistwidth</code>	<code>\glswrite</code>
218, 220–222	88
<code>\glspar</code>	<code>\glswriteentry</code>
108	7
<code>\GLSpl</code>	<code>\Glsxtr</code>
73, 85	35
<code>\Glspl</code>	<code>\Glsxtr</code>
34, 73, 85	35
<code>\glspl</code>	<code>\Glsxtr@@do@wrglossary</code>
34, 73, 85	7, 9, 10
<code>\GLSplural</code>	<code>\Glsxtr@addlocclstfield</code>
203, 204	9, 10
<code>\Glsplural</code>	<code>\Glsxtr@addunused</code>
204	30
<code>\glsplural</code>	<code>\Glsxtr@applyabbrvfmt</code>
203, 204	158
<code>\glspluralaccessdisplay</code>	<code>\Glsxtr@applyabbrvstyle</code>
111, 112	144, 145, 161
<code>\glspluralsuffix</code>	<code>\Glsxtr@counterrecord</code>
105, 145, 148	106
<code>\glspostdescription</code>	<code>\Glsxtr@doption</code>
139, 218–224	4, 11, 16
<code>\glspostinline</code>	<code>\Glsxtr@fields</code>
223	105
<code>\glspostlinkhook</code> .	<code>\Glsxtr@headentry@p</code>
40, 42, 52–56, 67, 149–158	69, 70
<code>\glsprestandardsort</code>	<code>\Glsxtr@ifnextpunc</code>
91	142
<code>\glsresetentrylist</code>	<code>\Glsxtr@ifpunctoken</code>
97, 107	142
<code>\glssee</code>	<code>\Glsxtr@indexonly@saveentrycounter</code>
23	9, 10, 17
<code>\glsseeformat</code>	<code>\Glsxtr@keylist</code>
30, 89, 94, 95	33, 34
<code>\glssetabbrvfmt</code>	<code>\Glsxtr@langtag</code>
40, 44, 65, 66, 126–128, 130, 131, 137, 149–153, 155, 156	105
	<code>\Glsxtr@linkprefix</code>
	105

<code>\glxtr@makeglossaries</code>	89	<code>\glxtrdowrglossaryhook</code>	61
<code>\glxtr@newabbreviation</code>	87, 145	<code>\GlsXtrEnableEntryCounting</code>	85
<code>\glxtr@next</code>	143	<code>\GlsXtrEnableEntryUnitCounting</code>	73
<code>\glxtr@org@getgrouptitle</code>	96	<code>\GlsXtrEnableOnTheFly</code>	33, 35
<code>\glxtr@org@newignoredglossary</code>	24	<code>\glxtrfieldlistgadd</code>	106
<code>\glxtr@org@makenoidxglossaries</code>	31	<code>\glxtrfieldtitlecase</code>	127–130
<code>\glxtr@pluralsuffixes</code>	105	<code>\glxtrfieldtitlecasecs</code>	126
<code>\glxtr@provideignoredglossary</code>	26	<code>\glxtrfieldxifinlist</code>	108
<code>\glxtr@punctlist</code>	142, 143	<code>\glxtrfirstscfont</code>	177–180
<code>\glxtr@record</code>	7	<code>\glxtrfirstsmfont</code>	180–183
<code>\glxtr@recordsee</code>	8	<code>\GlsXtrFmtDefaultOptions</code>	19
<code>\glxtr@resource</code>	104	<code>\GlsXtrFmtField</code>	19
<code>\glxtr@s@newignoredglossary</code>	24	<code>\GlsXtrFormatLocationList</code>	37, 39, 230–232
<code>\glxtr@s@provideignoredglossary</code>	26	<code>\GLSxtrfull</code>	13, 208
<code>\glxtr@saveentrycounter</code>	7, 8, 60	<code>\Glsxtrfull</code>	12, 208, 209
<code>\glxtr@setup@record</code>	9, 10, 16, 17	<code>\glxtrfull</code>	12, 208
<code>\glxtr@shortcutsval</code>	105	<code>\Glsxtrfullformat</code>	147, 160, 162, 165, 166, 169, 171, 172, 174, 175, 192, 193, 195, 197
<code>\glxtr@texencoding</code>	105	<code>\glxtrfullformat</code>	147, 160, 162, 164– 168, 170, 172, 174, 175, 191, 193, 195, 197
<code>\glxtr@usesee</code>	29	<code>\GLSxtrfullpl</code>	13, 208, 209
<code>\glxtr@warnonexistsordo</code>	6, 9, 10, 28, 29	<code>\Glsxtrfullpl</code>	12, 209
<code>\glxtr@writefields</code>	104	<code>\glxtrfullpl</code>	12, 208
<code>\glxtrabbrvfootnote</code>	168–170	<code>\Glsxtrfullplformat</code>	148, 160, 162, 165, 167, 169, 171, 173–175, 192, 193, 195, 197
<code>\glxtrabbrvpluralsuffix</code>	105, 149, 164, 166, 168, 170, 172, 173, 175, 176, 180, 191	<code>\glxtrfullplformat</code>	160, 162, 164, 166, 168, 170, 172, 174, 175, 191, 193, 195, 197
<code>\glxtrabbrvtype</code>	12, 146	<code>\glxtrfullsep</code>	147, 164–167, 169, 171–175, 184–186, 190
<code>\glxtraddallcrossrefs</code>	30	<code>\glxtrgenabbrvfmt</code>	40
<code>\glxtralias</code>	60	<code>\glxtrgetgrouptitle</code>	96
<code>\glxtrAltTreeIndent</code>	224	<code>\Glsxtrheadfirst</code>	200
<code>\glxtralttreeInit</code>	232	<code>\glxtrheadfirst</code>	200
<code>\glxtrAltTreePar</code>	223	<code>\Glsxtrheadfirstplural</code>	200
<code>\glxtrAltTreeSetHangIndent</code>	224, 233	<code>\glxtrheadfirstplural</code>	200
<code>\glxtrAltTreeSetSubHangIndent</code>	233	<code>\Glsxtrheadfull</code>	200
<code>\glxtralttreeSubSymbolDescLocation</code>	233	<code>\glxtrheadfull</code>	200
<code>\glxtralttreeSymbolDescLocation</code>	224, 233	<code>\Glsxtrheadfullpl</code>	200
<code>\glxtrassignfieldfont</code>	45–52	<code>\glxtrheadfullpl</code>	200
<code>\glxtrcat</code>	33, 34	<code>\glxtrheadlong</code>	200
<code>\glxtrchecknohyperfirst</code>	45–47	<code>\glxtrheadlong</code>	200
<code>\glxtrComputeTreeIndent</code>	233	<code>\Glsxtrheadlongpl</code>	200
<code>\glxtrComputeTreeSubIndent</code>	233	<code>\glxtrheadlongpl</code>	200
<code>\GlsXtrDefineAbbreviationShortcuts</code>	14	<code>\Glsxtrheadplural</code>	200
<code>\GlsXtrDefineOtherShortcuts</code>	14	<code>\glxtrheadplural</code>	200
<code>\glxtrdiscardperiod</code>	140	<code>\Glsxtrheadshort</code>	200
<code>\glxtrdisplayendloc</code>	97	<code>\glxtrheadshort</code>	200
<code>\glxtrdisplayendloohook</code>	98	<code>\Glsxtrheadshortpl</code>	200
<code>\glxtrdisplaysingleloc</code>	97, 98	<code>\glxtrheadshortpl</code>	200
<code>\glxtrdisplaystartloc</code>	97		
<code>\glxtrdoautoindexname</code>	60, 61, 132		
<code>\glxtrdopostpunc</code>	170		

<code>\Glsxtrheadtext</code>	200	<code>\Glsxtrp</code>	68
<code>\glsxtrheadtext</code>	200	<code>\glsxtrp</code>	68, 70
<code>\glsxtrifcounttrigger</code>	76, 77	<code>\Glsxtrpl</code>	35
<code>\glsxtrifemptyglossary</code>	97, 102, 107	<code>\glsxtrpl</code>	35
<code>\glsxtrifindexing</code>	60	<code>\glsxtrpostdescription</code>	125, 139, 223
<code>\glsxtrifinmark</code>	69–72, 199, 200	<code>\glsxtrpostlink</code>	140
<code>\glsxtrifnextpunc</code>	142, 143	<code>\glsxtrpostlinkendsentence</code>	140
<code>\glsxtrifperiod</code>	141, 142	<code>\glsxtrpostlinkhook</code>	140
<code>\glsxtrifwasfirstuse</code> ...	44–47, 52–56, 58, 88, 141, 149, 152–158, 170, 192, 194, 196	<code>\glsxtrpostlocalreset</code>	73, 75, 83
<code>\glsxtrindexaliased</code>	59, 60	<code>\glsxtrpostlocalunset</code>	73, 74, 82, 83
<code>\glsxtrinitwrgloss</code>	43	<code>\glsxtrpostlongdescription</code>	24
<code>\glsxtrinitwrglossbeforefalse</code>	42	<code>\glsxtrpostnamehook</code>	129–132
<code>\glsxtrinitwrglossbeforetrue</code>	42	<code>\GlsXtrPostNewAbbreviation</code>	
<code>\Glsxtrinlinefullformat</code> ...	148, 150, 162, 169, 171, 172, 174, 175, 193, 195, 214	146, 162, 164–168, 170, 172–174, 176, 185, 186, 188, 191, 192, 194, 196, 197
<code>\glsxtrinlinefullformat</code>	148– 150, 162, 169, 171–173, 175, 193, 195, 214	<code>\glsxtrpostreset</code>	73, 75, 83
<code>\Glsxtrinlinefullplformat</code> ..	148, 151, 162, 169, 171, 172, 174, 175, 193, 195, 215	<code>\glsxtrpostunset</code>	72, 74, 82
<code>\glsxtrinlinefullplformat</code> ..	147, 148, 151, 162, 169, 171–173, 175, 193, 195, 214	<code>\glsxtrprotectlinks</code>	62, 63
<code>\glsxtrininsertinsidefalse</code>	164	<code>\GlsXtrRecordCounter</code>	7
<code>\glsxtrlocationhyperlink</code>	98	<code>\glsxtrregularfont</code>	40, 44, 45
<code>\glsxtrlocrangefmt</code>	98	<code>\glsxtrresourcecount</code>	104
<code>\GLSxtrlong</code>	12, 206, 207	<code>\glsxtrresourcefile</code>	104
<code>\Glsxtrlong</code>	12, 207	<code>\glsxtrrestoremarkhook</code>	198, 199
<code>\glsxtrlong</code>	12, 206	<code>\glsxtrscfont</code>	176–180
<code>\GLSxtrlongpl</code>	12, 206, 207	<code>\glsxtrscsuffix</code>	177–180
<code>\Glsxtrlongpl</code>	12, 207	<code>\GlsXtrSetActualChar</code>	136
<code>\glsxtrlongpl</code>	12, 206	<code>\glsxtrsetaliasnoindex</code>	9, 10, 59, 60
<code>\glsxtrlongshortdescsort</code>	165	<code>\GlsXtrSetEncapChar</code>	136
<code>\glsxtrmarkhook</code>	198, 199	<code>\GlsXtrSetEscChar</code>	136
<code>\glsxtrnewabbrevpresetkeyhook</code>	146	<code>\glsxtrsetfieldifexists</code>	21, 22
<code>\glsxtrnewnumber</code>	13	<code>\GlsXtrSetLevelChar</code>	136
<code>\glsxtrnewsymbol</code>	13	<code>\glsxtrsetpopts</code>	67
<code>\glsxtrNoGlossaryWarning</code>	14, 15, 99	<code>\glsxtrsetupfulldefs</code>	149–151, 170
<code>\GlsXtrNoGlsWarningAutoMake</code>	103	<code>\GLSxtrshort</code>	12, 71, 72, 201, 202
<code>\GlsXtrNoGlsWarningBuildInfo</code>	103	<code>\Glsxtrshort</code>	12, 202
<code>\GlsXtrNoGlsWarningCheckFile</code>	103	<code>\glsxtrshort</code>	12, 201
<code>\GlsXtrNoGlsWarningEmptyMain</code>	103	<code>\GLSxtrshortpl</code>	12, 201, 202
<code>\GlsXtrNoGlsWarningEmptyNotMain</code> ...	103	<code>\Glsxtrshortpl</code>	12, 202
<code>\GlsXtrNoGlsWarningEmptyStart</code>	103	<code>\glsxtrshortpl</code>	12, 201, 202
<code>\GlsXtrNoGlsWarningHead</code>	102	<code>\glsxtrsmfont</code>	180–183
<code>\GlsXtrNoGlsWarningMisMatch</code>	103	<code>\glsxtrsmsuffix</code>	180–183
<code>\GlsXtrNoGlsWarningNoOut</code>	103	<code>\glsxtrsupplocationurl</code>	99
<code>\GlsXtrNoGlsWarningTail</code>	103	<code>\glsxtrtagfont</code>	139
<code>\glsxtrorg@ifKV@glslink@hyper</code>	40	<code>\Glsxtrtitlefirst</code>	199, 200, 212
<code>\GLSxtrp</code>	69	<code>\glsxtrtitlefirst</code>	199, 200, 212
		<code>\Glsxtrtitlefirstplural</code>	199–201, 212, 213
		<code>\glsxtrtitlefirstplural</code> ...	199–201, 212
		<code>\Glsxtrtitlefull</code>	200, 201, 214
		<code>\glsxtrtitlefull</code>	200, 201, 214

<code>\Glsxtrtitlefullpl</code>	200, 201, 215	<code>\ifcsstring</code>	17, 122, 161
<code>\glxstrtitlefullpl</code>	200, 201, 214, 215	<code>\ifcsundef</code>	22,
<code>\Glsxtrtitlelong</code>	200, 201, 213		23, 25–27, 31, 35, 38, 63, 74, 80–83, 95,
<code>\glxstrtitlelong</code>	199–201, 213		96, 99, 108, 122, 160, 162, 163, 217, 225, 232
<code>\Glsxtrtitlelongpl</code>	200, 201, 214	<code>\ifcsvoid</code>	23, 121
<code>\glxstrtitlelongpl</code>	199–201, 213	<code>\ifdef</code>	9, 13, 19,
<code>\Glsxtrtitleplural</code>	199, 200, 211		28, 29, 36, 37, 58, 59, 69–71, 94, 95, 105,
<code>\glxstrtitleplural</code>	199, 200, 211		124, 125, 136, 139, 190, 209–215, 218, 223
<code>\Glsxtrtitleshort</code>	199, 200, 210	<code>\ifdefempty</code>	6, 7, 25–27, 29,
<code>\glxstrtitleshort</code>	199, 200, 209		44, 74, 85, 86, 89, 91, 98, 107, 109, 137, 158
<code>\Glsxtrtitleshortpl</code>	199, 200, 210	<code>\ifdefequal</code>	103, 109
<code>\glxstrtitleshortpl</code>	199, 200, 210	<code>\ifdefstring</code>	5, 133, 138
<code>\Glsxtrtitletext</code>	199, 200, 211	<code>\ifdefvoid</code>	23, 29, 30, 63, 79, 99, 109
<code>\glxstrtitletext</code>	199, 200, 210, 211	<code>\ifdim</code>	36, 37, 88, 225–232
<code>\glxstrtreetopindent</code>	224, 232	<code>\IfFileExists</code>	15, 99, 103–105, 217
<code>\glxstrundefaction</code> ...	6, 9, 10, 17, 25, 27–29	<code>\ifglossaryexists</code>	29
<code>\glxstrundeftag</code>	17	<code>\ifglssacronym</code>	12, 103
<code>\glxstrunsrtdo</code>	108	<code>\ifglssacrshortcuts</code>	13
<code>\GlsXtrUseAbbrStyleFmts</code>		<code>\ifglssautomake</code>	91, 103, 105
	165, 167, 176–190, 194, 196, 198	<code>\ifglssentrycounter</code>	22
<code>\GlsXtrUseAbbrStyleSetup</code>	176–189, 196, 198	<code>\ifglssentryexists</code>	
<code>\glxstruserfield</code>	190		28, 33, 34, 37, 44, 122, 139, 140
<code>\glxstruserparen</code>	191–197	<code>\ifglssfieldeq</code>	121
<code>\glxstrusersuffix</code>	191, 192, 195, 197	<code>\ifglsshasfield</code>	19, 59, 190
<code>\glxstruseseeformat</code>	30	<code>\ifglsshaslong</code>	79
<code>\GlsXtrWarnDeprecatedAbbrStyle</code>	144, 163	<code>\ifglsshasparent</code>	109, 225–228
<code>\GlsXtrWarning</code>	33, 34	<code>\ifglsshasshort</code>	40, 44
H			
<code>\hangindent</code>	224, 232	<code>\ifglsshasymbol</code>	141, 224
<code>\hbox</code>	218	<code>\ifglssindexonlyfirst</code>	60
<code>\hfill</code>	218	<code>\ifglssnogroupskip</code>	219–223, 234
<code>\href</code>	62	<code>\ifglssnonumberlist</code>	39
<code>\hsize</code>	36, 37	<code>\ifglssanitizesort</code>	91
<code>\hss</code>	218	<code>\ifglssubentrycounter</code>	22
<code>\hyperlink</code>	63, 99	<code>\ifglssused</code>	30, 31, 58,
<code>\hyperpage</code>	132		60, 75, 84, 88, 158, 225, 226, 228, 230, 231
<code>\hyperref</code>	62, 99	<code>\ifglssxindy</code>	99, 101
<code>hyperref package</code>	63, 132, 198, 209	<code>\ifglssxtrinitwrglossbefore</code>	42, 43
I			
<code>\if</code>	32	<code>\ifglssxtrinertinside</code>	
<code>\if@glxstr@format@override</code>	133		152–158, 164–175, 191–193, 195, 197
<code>\if@glxstrdocdefrestricted</code>	31	<code>\ifHy@hyperindex</code>	132
<code>\if@glxstrindexcrossrefs</code>	11, 30	<code>\ifinlistcs</code>	20, 31
<code>\ifblank</code>	15, 18, 33, 34, 88	<code>\ifKV@glsslink@hyper</code>	40, 43
<code>\ifcase</code>	6, 9, 13, 14, 31, 42, 93	<code>\ifKV@glsslink@local</code>	42
<code>\ifcsdef</code>	6, 17, 22, 23, 25–	<code>\ifKV@glsslink@noindex</code>	7, 8, 60
	27, 56, 57, 68–72, 80, 92, 96, 108, 127,	<code>\ifnum</code>	10, 11, 76, 84, 95, 104, 233
	128, 130, 131, 140, 144, 158, 161, 218–222	<code>\ifthenelse</code>	103
		<code>\IfTrackedLanguageFileExists</code>	215
		<code>\ifundef</code>	63, 88, 138, 139

<code>\ifx</code>	36, 37, 92, 97, 98, 106, 133, 134, 136, 143, 144, 234	<code>makeindex</code>	88
<code>\immediate</code>	75, 84, 99, 100, 105	<code>\makenoidxglossaries</code>	101
<code>\index</code>	133	<code>\MakeTextUppercase</code>	200
<code>\indexspace</code>	234	<code>\MakeUppercase</code>	199, 200
<code>\input</code>	215	<code>\markboth</code>	199
<code>\inputencodingname</code>	105	<code>\markright</code>	199
<code>\istfilename</code>	89	<code>\maxdimen</code>	36, 37
<code>\item</code>	101, 102, 218	<code>\mbox</code>	233
J		<code>\medskip</code>	103, 108
<code>\jobname</code>	99, 101–105	<code>\MessageBreak</code>	32, 35, 76, 85, 91, 92, 161
K		<code>mfistuc package</code>	138
<code>\key@ifundefined</code>	8, 9, 18, 56, 107, 109	<code>\mfistucMakeUppercase</code> .	45–56, 58, 65– 67, 69, 71, 72, 78, 79, 86, 87, 111–120, 129, 130, 150, 151, 153, 155, 156, 158–160
<code>\KV@glslink@hyperfalse</code> ..	45, 58, 59, 63, 64	<code>\mfu@checkword@arg</code>	138
<code>\KV@glslink@noindexfalse</code>	59	<code>\mfu@checkword@do</code>	138
<code>\KV@glslink@noindextrue</code>	59, 64	N	
L		<code>\NeedsTeXFormat</code>	4, 217
<code>\LaTeX</code>	101, 102	<code>\new@glossaryentry</code>	32, 91
<code>\leaders</code>	218	<code>\new@ifnextchar</code>	57, 78, 142, 149–158
<code>\leavevmode</code>	24, 43	<code>\newabbr</code>	13
<code>\let</code>	4, 6, 7, 9–13, 16, 17, 19, 24, 31, 32, 35, 36, 38–64, 67, 73–75, 82–96, 104–107, 109, 127–134, 138, 139, 142–145, 149–158, 170, 198–201, 223, 225	<code>\newabbreviation</code>	13
<code>\letabbreviationstyle</code>	169, 171, 173, 174, 176, 178, 182, 187	<code>\newabbreviationhook</code>	146
<code>\letcs</code> .	18, 29, 30, 56, 94–96, 108, 109, 127–131	<code>\newabbreviationstyle</code>	164–169, 171, 173, 174, 176–189, 191–194, 196, 198
<code>\levelchar</code>	136	<code>\newacronym</code>	86, 87
<code>\listadd</code>	80	<code>\newacronymhook</code>	86
<code>\listbreak</code>	138	<code>\newacronymstyle</code>	87
<code>\listcsadd</code>	20	<code>\newcommand</code> .	4–8, 10–35, 37–40, 42, 44, 45, 56, 57, 59–62, 64, 67, 69–74, 76, 78–82, 84, 85, 87, 88, 92–102, 104–126, 132– 158, 160–163, 165, 167, 168, 176, 180, 184, 190, 191, 199–215, 217, 223–225, 232
<code>\listcseadd</code>	20, 81	<code>\newcount</code>	10, 104
<code>\listcsgadd</code>	20, 31	<code>\newentry</code>	13
<code>\listcsxadd</code>	20, 80	<code>\newglossary</code>	12, 89
<code>\loadglsentries</code>	32, 101	<code>\newglossaryentry</code>	13, 31, 32, 74, 82, 86, 124, 125, 146
<code>\long</code>	24	<code>\newglossaryentry options</code>	
M		alias	23
<code>\MakeAcronymsAbbreviations</code>	87	desc	114, 119
<code>\makeatletter</code>	99, 104, 135	descplural	115, 119
<code>\makeatother</code>	135	first	62, 112, 118, 163, 204–206, 211, 235
<code>\makebox</code>	218, 233	firstplural ..	112, 113, 118, 163, 205, 212, 235
<code>\makefirstuc</code>	138	loclist	20
<code>makeglossaries</code>	93	long	117, 120, 213
<code>\makeglossaries</code>	88, 100–103, 106	longplural	117, 120, 213
<code>\makeglossary</code>	89	name	110, 117, 118, 133
<code>makeindex</code>	235		

type	91		S	
\printnoidxglossaries	102		\s@glshyp@opt	61
\printnoidxglossary	90, 102		\s@glsxtr@enabletagging	137
\printnumbers	13, 125		\s@printunsrtglossary	106, 108
\printsymbols	13, 125		\seename	30
\printunsrtglossary	107		\setabbreviationstyle	87, 165, 173
\printunsrtglossaryhandler	107, 108		\setacronymstyle	87, 88
\printunsrtglossaryunit	9, 10, 108		\setentrycounter	97
\printunsrtglossaryunitsetup	108		\SetGenericNewAcronym	87
\ProcessOptions	217		\setglossarystyle	16, 92, 218, 234
\ProcessOptionsX	16		\setkeys	6, 16, 43, 44, 60, 86, 92, 145, 146
\protect	69–72, 118–120, 147, 164–186, 188, 191–194, 196, 197, 201–209		\setlength	36, 37, 224, 233
\protected@csedef	22, 224		\settowidth	88, 224–232
\protected@csxdef	22, 224		\setupglossaries	4, 16
\protected@edef .	36, 86, 95, 96, 108, 133, 146		\sfcode	140, 141, 223
\protected@write ...	7, 8, 39, 88, 89, 104–106		\space	5, 7, 33, 35, 60, 73–76, 82, 84, 85, 87–90, 92, 100, 103, 106, 108, 141, 147, 165, 223, 224, 232
\protected@xdef	109		\spacefactor	140, 141, 145, 223
\providecommand	12, 18, 39, 58, 59, 75, 84, 88, 89, 99, 100, 217		\string	5, 7, 8, 32, 33, 35, 39, 56, 57, 60, 68, 69, 73–76, 82, 84, 85, 87–92, 99–106, 108, 128, 130–133
\ProvidesFile	215		\strut	218–223
\ProvidesPackage	4, 217		\subglossentry	92, 109, 218–223, 233
		Q		
\quotechar	136			
		R		T
\raggedright	220, 222		\tablehead	221, 222
\relax	6, 9, 10, 13, 14, 16, 19, 32, 35–38, 42, 61, 67, 75, 76, 84, 89, 90, 92–95, 97, 104–106, 109, 134, 136, 138, 140, 141, 144, 145, 225–234		\tabletail	221, 222
relsize package	180		\tabularnewline	218–223
\renewcommand	5, 9–16, 23–29, 31, 32, 35, 37–42, 56, 58– 60, 62, 63, 67, 72–75, 79, 82–93, 95–97, 99, 100, 108, 124, 126–129, 131, 136– 140, 148, 162, 164–199, 218–223, 233, 234		\TeX	101
\renewenvironment	218–222, 232		\texorpdfstring	19, 69–71, 209–215
\renewglossarystyle	218–222, 232		textcase package	198
\renewrobustcmd	44, 63		\textsc	176
\RequireGlossariesExtraLang	216		\textsmaller	180
\RequirePackage	4, 14–16, 217		\texttt	100–103
\reserved@a	142		\the ...	86, 98, 104, 136, 146, 164–168, 170– 174, 176, 184–186, 188, 191–194, 196, 197
\reserved@b	142		\theglsentrycounter	7, 44
\reserved@d	143		\theHglentrycounter	7, 44
\RestoreAcronyms	87		\theindex	132
\romannumeral	224, 225, 232		\this@dialect	215
			\toks@	98, 136
			tracklang package	105
		U		
		\undef		9, 10, 137
		\underline		139
		\unskip		24, 31, 218
		\usepackage		102, 103

V		\xdef	92
\val	6, 9, 10, 13, 14, 42, 93	\xifinlist	80
W		\xifinlistcs	21
\warn@nomakeglossaries	90	xindy	235
\warn@noprintglossary	90, 93	xindy	88
\write	75, 84, 89, 99, 100, 105	xkeyval package	4
X		\XKV@checkchoice	39
\x	98	\XKV@plfalse	39
\xcapitalisewords	126	\XKV@resa	39
		\XKV@sttrue	39