glossaries-extra.sty v1.52: an extension to the glossaries package

Nicola L.C. Talbot
Dickimaw Books
dickimaw-books.com

2023-06-28

This document is also available as HTML (glossaries-extra-manual.html).

Abstract

The glossaries-extra package is an extension to the glossaries package, providing additional features. In particular, it provides a completely different abbreviation mechanism. You will need at least glossaries version 4.19, but it is best to update both packages at the same time, if new releases are available for both of them.

The glossaries-extra package uses a different set of defaults to the base glossaries package. This means that if you simply replace glossaries with glossaries-extra in an existing document, there may be some differences in the PDF, and you may encounter errors. See §1.1 for more details.

This document assumes some familiarity with the glossaries package. If you are new to glossaries, you may prefer to start with the following:

- The glossaries package: a guide for beginners
texdoc glossariesbegin

- glossaries-extra and bib2gls: an introductory guide
textb bib2gls-begin
## Contents

List of Examples ........................................ vii

### I. User Guide ........................................ 1

#### 1. Introduction ................................... 2

1.1. Package Defaults .................................. 2
1.2. Example Differences Between glossaries and glossaries-extra .......... 3
   1.2.1. Basic defaults ................................ 3
   1.2.2. Language defaults ............................. 4
   1.2.3. Combined with memoir .......................... 4
   1.2.4. Abbreviations .................................. 5
   1.2.5. Glossary Mid-Build Placeholder (\printglossary) ................. 5
1.3. Further Reading ................................... 7

#### 2. Package Options ................................. 8

2.1. Glossary Lists ................................... 9
2.2. Glossary Style Options ............................. 12
2.3. Loading Other Packages ............................. 14
2.4. Entry Definitions, References and Indexing ......................... 15
2.5. Debugging ....................................... 28

#### 3. Defining Entries ................................. 32

3.1. Command Definitions ................................ 32
3.2. Glossary Entry Keys ................................ 33
3.3. Plurals ........................................... 35
3.4. Entry Aliases ...................................... 36
3.5. Setting or Updating Fields ........................... 37

#### 4. Abbreviations ................................... 41

4.1. Defining Abbreviations .............................. 41
   4.1.1. Abbreviation Fields: long and short ............... 43
   4.1.2. Abbreviation Fields: longplural and shortplural .... 43
   4.1.3. Abbreviation Fields: name and description ........... 44
   4.1.4. Abbreviation Fields: type .......................... 44
   4.1.5. General Hooks ................................. 45
4.2. Examples: makeindex vs bib2gls ...................... 45
Contents

4.3. Referencing (Using) Abbreviations ................................. 47
  4.3.1. Prefixes ...................................................... 54
  4.3.2. Abbreviation Shortcut Commands ......................... 56
4.4. Tagging Initials .................................................. 56
4.5. Abbreviation Styles ............................................. 59
  4.5.1. Predefined Abbreviation Styles ......................... 60
  4.5.2. Advanced Style Commands ............................... 159
  4.5.3. Defining New Abbreviation Styles ..................... 161
4.6. Restoring Base Acronym Mechanism .............................. 181

5. Referencing (Using) Entries 183
  5.1. Options ......................................................... 186
    5.1.1. Setting Up Defaults .................................. 187
    5.1.2. Additional Options .................................... 190
  5.2. Case Changing ................................................ 197
    5.2.1. Sentence Case Commands ............................... 197
    5.2.2. Lower Case ............................................. 200
    5.2.3. Upper Case ............................................. 200
    5.2.4. Title Case ............................................. 200
  5.3. Entries in Sectioning Titles, Headers, Captions and Contents 201
    5.3.1. Simplistic Approach .................................. 202
    5.3.2. New Commands Designed for Chapter/Section Headings or Captions 204
    5.3.3. Advanced Commands .................................... 214
  5.4. Nested Links .................................................. 228
  5.5. Adjusting the Text Style .................................... 235
    5.5.1. Outer Formatting ....................................... 239
    5.5.2. Middle Formatting ...................................... 240
    5.5.3. Inner Formatting ....................................... 242
    5.5.4. Post Link Hook ......................................... 246
    5.5.5. Entry Format Style ..................................... 255
  5.6. Hyperlinks ..................................................... 257
  5.7. Label Prefixes ................................................ 258
  5.8. Indexing ......................................................... 260
  5.9. Cross-Referencing .......................................... 269
    5.9.1. Entries that may not be required .................... 271
    5.9.2. Accessing the Cross-Referencing Fields ............ 279
    5.9.3. Cross-Reference Indexing ............................. 281
  5.10. First Use Flag .............................................. 282
    5.10.1. Buffering Unsets ..................................... 287
  5.11. Accessing Fields ............................................ 294
  5.12. Encapsulation (Formatting) Based on Field Values ....... 296
    5.12.1. Foreign Language Field ............................. 296
    5.12.2. Associated Entry Format ............................ 298
  5.13. Comma-Separated Lists ...................................... 302
Contents

5.14. List Fields .......................................................... 307
5.15. Field Conditionals .................................................. 308

6. Counting References .................................................... 310
  6.1. Entry Counting (First Use Flag) ................................. 310
  6.2. Link Counting ...................................................... 320

7. Multi (or Compound) Entries ........................................... 325
  7.1. Examples ............................................................ 329
    7.1.1. Example: Hierarchical ...................................... 329
    7.1.2. Example: Suffix .............................................. 330
    7.1.3. Example: Category Suffix ................................... 331
    7.1.4. Example: Separators ........................................ 332
    7.1.5. Example: Skipping Elements (Fragment Element) ........ 333
    7.1.6. Example: Skipping Elements (Prefix and Post-Link Hooks) .... 336
  7.2. Main and Other Elements ......................................... 342
  7.3. Prefixes and Suffixes ............................................. 343
  7.4. Separators ......................................................... 346
  7.5. \gls Element Hooks .............................................. 350
  7.6. Post-Link Hook ................................................... 352
    7.6.1. Last Element ............................................... 353
    7.6.2. Main Element ............................................... 354
  7.7. Multi-Entry First Use ............................................ 355
  7.8. Multi-Entry Category ............................................. 356
  7.9. Multi-Entry Settings ............................................. 357
    7.9.1. Indexing ...................................................... 358
    7.9.2. Location Formats (Encaps) ................................. 358
    7.9.3. Post-Link Hooks ............................................. 359
    7.9.4. Prefixes and Suffixes ...................................... 360
    7.9.5. Skipping Elements ......................................... 360
    7.9.6. General ...................................................... 361
  7.10. \gls Options ....................................................... 362
  7.11. Variants of \gls ................................................ 366
    7.11.1. \gls-like .................................................. 366
    7.11.2. Abbreviations ............................................ 367
    7.11.3. Other Fields .............................................. 368
    7.11.4. Support for glossaries-prefix (\pgls) .................... 370
  7.12. Cross-References ................................................ 372
  7.13. Additional Commands ............................................ 373
  7.14. bib2gls .......................................................... 375

8. Defining and Displaying Glossaries ................................ 378
  8.1. Entry Page Reference ............................................ 380
  8.2. Glossary Preamble ............................................... 380
## Contents

11.5.8. Supplementary Commands ........................................ 597

12. Auto-Indexing ......................................................... 600

13. On-the-Fly Document Definitions .................................. 604

14. Supplementary Files .................................................. 607
   14.1. Dummy Files for Testing ........................................ 607
   14.2. Sample Files ..................................................... 608

15. Multi-Lingual Support ................................................ 618

II. Summaries and Index .................................................. 623

Symbols ................................................................. 624

Terms ................................................................. 625

Glossary Entry Keys Summary .......................................... 631

Glossary Entry Fields Summary ........................................ 640

\gls-Like and \glstext-Like Options Summary ....................... 642

Multi-Entry Set Options Summary .................................... 645

Print [Unsrt|noidx] Glossary Options Summary ..................... 650

Abbreviation Styles Summary .......................................... 654

Glossary Styles Summary .............................................. 671

Command Summary ..................................................... 680

Command Summary: @ ................................................. 680
Command Summary: A ................................................. 681
Command Summary: B ................................................. 691
Command Summary: C ................................................. 692
Command Summary: D ................................................. 696
Command Summary: E ................................................. 700
Command Summary: F ................................................. 701
Command Summary: G ................................................. 702
Command Summary: Glo ................................................. 702
Command Summary: Gls ................................................. 706
Command Summary: Glsxtr ........................................... 819
Command Summary: H ................................................. 934
List of Examples

1. Multiple abbreviation styles .............................................. 42
2. \newabbreviation vs \newacronym vs \newglossaryentry .................. 46
3. \@abbreviation vs \@acronym vs \@entry .................................. 47
4. Referencing an abbreviation (with hyperref) .......................... 49
5. First use of \gls vs \glsxtrfull vs \glsfirst ......................... 51
6. Tagging abbreviation initials ............................................. 59
7. Category without an associated abbreviation style .................... 60
8. The short-nolong abbreviation style .................................... 65
9. The short-nolong-desc abbreviation style ............................... 66
10. The nolong-short abbreviation style .................................... 67
11. The short-sc-nolong abbreviation style ................................ 68
12. The short-sc-nolong-desc abbreviation style ........................... 68
13. The nolong-short-sc abbreviation style ................................. 69
14. The short-sm-nolong abbreviation style ................................ 70
15. The short-sm-nolong-desc abbreviation style .......................... 70
16. The nolong-short-sm abbreviation style ................................ 71
17. The short-em-nolong abbreviation style ................................ 72
18. The short-em-nolong-desc abbreviation style ......................... 72
19. The nolong-short-em abbreviation style ................................ 73
20. The long-noshort-desc abbreviation style ............................... 74
21. The long-noshort abbreviation style .................................... 75
22. The long-noshort-sc abbreviation style ................................ 76
23. The long-noshort-sc-desc abbreviation style ........................... 76
24. The long-noshort-sm abbreviation style ................................ 77
25. The long-noshort-sm-desc abbreviation style ........................... 78
26. The long-noshort-em abbreviation style ................................ 78
27. The long-noshort-em-desc abbreviation style ........................... 79
28. The long-em-noshort-em abbreviation style ............................ 80
29. The long-em-noshort-em-desc abbreviation style ...................... 80
30. The long-short abbreviation style ...................................... 81
31. The long-short-desc abbreviation style ................................ 82
32. The long-short-sc abbreviation style .................................... 83
33. The long-short-sc-desc abbreviation style ............................... 84
34. The long-short-sm abbreviation style .................................... 84
35. The long-short-sm-desc abbreviation style ............................... 85
36. The long-short-em abbreviation style .................................... 86
List of Examples

37. The long-short-em-desc abbreviation style ........................................ 86
38. The long-em-short-em abbreviation style ........................................ 87
39. The long-em-short-em-desc abbreviation style ................................ 88
40. The long-short-user abbreviation style ........................................... 89
41. The long-short-user-desc abbreviation style .................................... 89
42. The long-postshort-user abbreviation style ..................................... 90
43. The long-postshort-user-desc abbreviation style ................................ 91
44. The long-postshort-sc-user abbreviation style .................................. 92
45. The long-postshort-sc-user-desc abbreviation style .............................. 92
46. The short-long abbreviation style ................................................ 93
47. The short-long-desc abbreviation style ........................................ 94
48. The short-sc-long abbreviation style ............................................... 95
49. The short-sc-long-desc abbreviation style ....................................... 95
50. The short-sm-long abbreviation style ............................................. 96
51. The short-sm-long-desc abbreviation style ....................................... 97
52. The short-em-long abbreviation style ............................................. 97
53. The short-em-long-desc abbreviation style ....................................... 98
54. The short-em-long-em abbreviation style ......................................... 99
55. The short-em-long-em-desc abbreviation style ................................... 99
56. The short-long-user abbreviation style ........................................... 100
57. The short-long-user-desc abbreviation style ..................................... 101
58. The short-postlong-user abbreviation style ...................................... 102
59. The short-postlong-user-desc abbreviation style ................................. 103
60. The long-hyphen-short-hyphen abbreviation style .............................. 104
61. The long-hyphen-postshort-hyphen abbreviation style ........................ 105
62. The long-hyphen-short-hyphen-desc abbreviation style ......................... 106
63. The long-hyphen-postshort-hyphen-desc abbreviation style .................. 106
64. The long-hyphen-noshort-desc-noreg abbreviation style ....................... 107
65. The long-hyphen-noshort-noreg abbreviation style .............................. 108
66. The short-hyphen-long-hyphen abbreviation style ................................ 109
67. The short-hyphen-postlong-hyphen abbreviation style .......................... 109
68. The short-hyphen-long-hyphen-desc abbreviation style ........................ 110
69. The short-hyphen-postlong-hyphen-desc abbreviation style .................. 111
70. The long-only-short-only abbreviation style ..................................... 112
71. The long-only-short-only-desc abbreviation style .............................. 112
72. The long-only-short-sc-only abbreviation style .................................. 113
73. The long-only-short-sc-only-desc abbreviation style .......................... 114
74. The short-footnote abbreviation style .......................................... 115
75. The short-footnote-desc abbreviation style ..................................... 116
76. The short-postfootnote abbreviation style ...................................... 117
77. The short-postfootnote-desc abbreviation style ................................ 118
78. The short-sc-footnote abbreviation style ........................................ 119
79. The short-sc-footnote-desc abbreviation style .................................. 120
80. The short-sc-postfootnote abbreviation style ................................... 121
List of Examples

81. The short-sc-postfootnote-desc abbreviation style ............... 122
82. The short-sm-footnote abbreviation style ......................... 123
83. The short-sm-footnote-desc abbreviation style .................. 124
84. The short-sm-postfootnote abbreviation style .................... 125
85. The short-sm-postfootnote-desc abbreviation style .............. 126
86. The short-em-footnote abbreviation style ......................... 127
87. The short-em-footnote-desc abbreviation style .................. 128
88. The short-em-postfootnote abbreviation style .................... 129
89. The short-em-postfootnote-desc abbreviation style .............. 130
90. Illustrating the prere-set option .................................. 192
91. Combining prere-set and pre-unset .................................. 193
92. References in section headings (simplistic approach) ............ 203
93. References in section headings using \glstxttext .................. 205
94. Reference with hyperlink in section headings ..................... 206
95. Nested link text with \glspct ........................................ 234
96. Link text styles: outer, middle, inner, hyperlinks and post-link hook ........ 237
97. Link text styles: outer, middle, inner, hyperlinks and post-link hooks (custom and abbreviation style) .................. 239
98. Changing the outer text format ...................................... 240
99. Middle formatting ...................................................... 241
100. Inner formatting ..................................................... 243
101. Category post-link hook .............................................. 247
102. Category post-link hook with punctuation lookahead ............ 250
103. Category post-link hooks ............................................. 253
104. Location list ordering (makeindex) ................................ 266
105. Location list ordering (bib2gls) ................................... 267
106. Cross-references (autoseeindex=true) ............................ 272
107. Cross-references (autoseeindex=false) ............................ 273
108. Cross-references (autoseeindex=false and post-name hook) ...... 274
109. Cross-references (no see, seealso or alias) ....................... 275
110. Cross-references (bib2gls) ........................................... 277
111. Cross-references (bib2gls and selection=recorded and deps and see) 278
112. Cross-references (bib2gls and selection=recorded and deps and see, prune-xr) ................................................. 279
113. Resetting the first use flag (\glssreset) ............................ 285
114. Local unset ............................................................ 286
115. Abbreviations with beamer (unset buffering) ....................... 291
116. Buffering first use unsets with \mbox ................................. 292
117. Alternatives to buffering ............................................. 293
118. Foreign language field encapsulation .............................. 298
119. Storing a formatting command in a field ........................... 301
120. Formatting lists contained in field values ......................... 306
121. Entry counting according to category ................................ 315
122. Entry unit counting (per section) according to category ........ 318
List of Examples

123. Enabling unit counting to hook into hyperlink setting ................. 320
124. Link counting used to selectively suppress hyperlinks ................ 323
125. Multi-entries: hierarchical ........................................... 330
126. Multi-entries: hierarchical with first-use suffix ....................... 331
127. Multi-entries: hierarchical with category suffix ....................... 332
128. Multi-entries: separators ............................................... 333
129. Multi-entries: skipping elements ...................................... 335
130. Multi-entries: skipping elements (unseting others) .................... 336
131. Multi-entries: skipping elements (prefix and post-link hooks) ...... 340
132. Changing the target prefix ............................................. 384
133. Prepending to the target prefix for just the entry item ............... 385
134. Displaying unsorted glossaries ....................................... 388
135. Displaying unsorted glossaries with stylemods ....................... 388
136. Displaying unsorted glossaries with different group settings ......... 390
137. Displaying unsorted glossaries with a copied list .................... 392
138. Displaying unsorted glossaries with custom groups ................... 396
139. Displaying unsorted glossaries with custom groups and sub-group headings 397
140. Displaying sorted glossaries with groups using bib2gls ............... 401
141. Filtering by category ................................................... 408
142. Inner glossaries using printunsrtglossarywrap ........................ 410
143. Nested glossaries ...................................................... 412
144. Sub-glossary for a given counter value ................................ 417
145. Sub-glossary for a given counter value ordered by use in the section 418
146. The bookindex style ................................................... 444
147. The topic style .......................................................... 476
148. The topicmcols style .................................................... 477
149. The topicmcols style with the widest name set ....................... 478
150. Two entries per row with \printunsrttable ............................ 486
151. Using bib2gls: one resource set .................................... 541
152. Using bib2gls: multiple resource sets ................................ 543
153. Using bib2gls: sub-blocks .............................................. 545
154. Using bib2gls: record counting ...................................... 553
155. Using bib2gls: unit record counting .................................. 556
156. Using bib2gls: unit record counting mini-glossary .................... 558
157. Using bib2gls: simple custom sort rule ................................ 564
158. Using bib2gls: dual backlinks ....................................... 588
159. Using bib2gls: dual entry label prefixes ................................ 596
Part I.

User Guide
1. Introduction

The glossaries package is a flexible package, but it’s also a heavy-weight package that uses a lot of resources. As package developer, I’m caught between those users who complain about the drawbacks of a heavy-weight package with a large user manual and those users who want more features (which necessarily adds to the package weight and manual size).

The glossaries-extra package is an attempt to provide a compromise for this conflict. Version 4.22 of the glossaries package is the last version to incorporate any major new features. Future versions of glossaries will mostly just be bug fixes. New features will instead be added to glossaries-extra. This means that the base glossaries package won’t increase in terms of package loading time and allocation of resources, but those users who do want extra features available will have more of a chance of getting their feature requests accepted.

Some of the commands provided by the base glossaries package are incompatible with glossaries-extra. These are marked with ⚠️ in this document.

The glossaries-extra package internally loads the glossaries package. As a general rule, it’s better to defer loading the base glossaries package to glossaries-extra rather than loading the two packages separately.

1.1. Package Defaults

I’m not happy with some of the default settings assumed by the glossaries package, and, judging from code I’ve seen, other users also seem unhappy with them, as certain package options are often used in questions posted on various sites. I can’t change the default behaviour of glossaries as it would break backward compatibility, but since glossaries-extra is a separate package, I have decided to implement some of these commonly-used options by default. You can switch them back if they’re not appropriate.

The new defaults are:

- **toc=true** (add the glossaries to the table of contents). Use **toc=false** to switch this back off.

- **nopostdot=true** (suppress the terminating full stop after the description in the glossary). Use **nopostdot=false** or just **postdot** to restore the terminating full stop. Alternatively, if you are interested in switching to bib2gls, you can instruct bib2gls to insert it with the **post-description-dot** option.
1. Introduction

- **noredefwarn** (suppress the warnings that occur when the `theglossary` environment and `\printglossary` are redefined while `glossaries` is loading). Note that this won’t have any effect if the `glossaries` package has already been loaded before you load the `glossaries-extra` package.

- If babel has been loaded, the `translate=babel` option is switched on. To revert to using the translator interface, use `translate=true`. There is no change to the default if babel hasn’t been loaded.

- The default style used by `\newacronym` is **short-nolong**. (That is, the long form is not shown on first use.) To revert back to “⟨long⟩ ⟨short⟩” on first use do:

  \[\setabbreviationstyle[acronym]{long-short}\]

  In the above example, **long-short** refers to the `glossaries-extra` abbreviation style not the `glossaries` acronym style of the same name. See §4 for further details.

1.2. Example Differences Between glossaries and glossaries-extra

The examples below illustrate the difference in explicit package options between `glossaries` and `glossaries-extra`. There may be other differences resulting from modifications to commands provided by `glossaries`.

1.2.1. Basic defaults

\[\documentclass{article}\]
\[\usepackage{glossaries-extra}\]

This is like:

\[\documentclass{article}\]
\[\usepackage[noheader,postdot]{glossaries}\]
\[\usepackage{glossaries-extra}\]
1. Introduction

1.2.2. Language defaults

\documentclass[british]{article}
\usepackage{babel}
\usepackage{glossaries-extra}

This is like:

\documentclass[british]{article}
\usepackage{babel}
\usepackage[\textit{toc},\textit{nopostdot},\textit{translate=babel}]{glossaries}
\usepackage{glossaries-extra}

1.2.3. Combined with memoir

\documentclass{memoir}
\usepackage{glossaries-extra}

This is like:

\documentclass{memoir}
\usepackage[\textit{toc},\textit{nopostdot},\textit{noredefwarn}]{glossaries}
\usepackage{glossaries-extra}

However

\documentclass{memoir}
\usepackage{glossaries}
\usepackage{glossaries-extra}

This is like:

\documentclass{memoir}
\usepackage[\textit{toc=false},\textit{nopostdot=false}]{glossaries}
\usepackage{glossaries-extra}
1. Introduction

Since by the time glossaries-extra has been loaded, the base glossaries package has already redefined memoir’s glossary-related commands.

1.2.4. Abbreviations

Abbreviations are defined with \newabbreviation:

\usepackage{glossaries-extra}
\newabbreviation{svm}{SVM}{support vector machine}
\begin{document}
First use: \gls{svm}. Explicit full form: \glsxtrfull{svm}.
\end{document}

This is the closest match to:

\usepackage{glossaries}
\newacronym{svm}{SVM}{support vector machine}
\begin{document}
First use: \gls{svm}. Explicit full form: \acrfull{svm}.
\end{document}

If you want to continue using \newacronym then you will need to change the style for the acronym category:

\usepackage{glossaries-extra}
\setabbreviationstyle[acronym]{long-short}
\newacronym{svm}{SVM}{support vector machine}
\begin{document}
First use: \gls{svm}. Explicit full form: \glsxtrfull{svm}.
\end{document}

Don’t use commands like \glstext or \glsfirst with abbreviations. See §4 for further details.

1.2.5. Glossary Mid-Build Placeholder (\printglossary)

Another noticeable change with glossaries-extra is that by default \printglossary will now display information text in the document if the external glossary file doesn’t exist. This is explanatory text to help new users who can’t work out what to do next to complete the
1. Introduction

document build. Once the document is set up correctly and the external files have been generated, this text will disappear.

This change is mostly likely to be noticed by users with one or more redundant empty glossaries who ignore transcript messages, explicitly use makeindex/xindy on just the non-empty glossary (or glossaries) and use the iterative \printglossaries command instead of \printglossary. For example, consider the following:

\documentclass{article}
\usepackage[acronym]{glossaries}
\makeglossaries
\newacronym{laser}{laser}{light amplification by stimulated emission of radiation}
\begin{document}
\gls{laser}
\printglossaries
\end{document}

The above document will only display the list of acronyms at the place where \printglossaries occurs. However it will also attempt to input the gls file associated with the main glossary.

If you use makeglossaries, you’ll get the warning message:

Warning: File 'test.glo' is empty.
Have you used any entries defined in glossary 'main'?
Remember to use package option 'nomain' if you don't want to use the main glossary.

(where the original file is called test.tex) but if you simply call makeindex directly to generate the acr file (without attempting to create the gls file) then the transcript file will always contain the message:

No file test.gls.

This doesn’t occur with makeglossaries as it will create the gls file containing the single command \null.

If you simply change from glossaries to glossaries-extra in this document, you’ll find a change in the resulting PDF if you don’t use makeglossaries and you only generate the acr file with makeindex.

The transcript file will still contain the message about the missing gls, but now you’ll also see information in the actual PDF document. The simplest remedy is to follow the advice inserted into the document at that point, which is to add the nomain package option:
1. Introduction

\documentclass{article}
\usepackage[nomain, acronym, postdot]{glossaries-extra}
\makeglossaries
\setabbreviationstyle[acronym]{long-short}
\newacronym{laser}{laser}{light amplification by stimulated emission of radiation}
\begin{document}
\gls{laser}
\printglossaries
\end{document}

Note the need to set the acronym style using \setabbreviationstyle before \newacronym. See §4 for further details.

1.3. Further Reading

The following documents and web pages are also available:

- The glossaries-extra documented code

  \texttt{texdoc\ glossaries-extra-code}

- Gallery: glossaries, glossaries-extra and bib2gls.\footnote{\url{dickimaw-books.com/gallery}}
- FAQs: glossaries, glossaries-extra and bib2gls.\footnote{\url{dickimaw-books.com/faq.php}}
- Incorporating \texttt{makeglossaries} or \texttt{makeglossaries-lite} or \texttt{bib2gls} into the document build.\footnote{\url{dickimaw-books.com/latex/buildglossaries/}}
- The \texttt{bib2gls} application.\footnote{\url{ctan.org/pkg/bib2gls}}
- The glossaries package.\footnote{\url{ctan.org/pkg/glossaries}}
This chapter describes the package options provided by glossaries-extra that are either not defined by the base glossaries package or are modified by glossaries-extra. You can additionally pass the base package options to glossaries-extra. For example, instead of:

\usepackage[nonumberlist]{glossaries}
\usepackage[abbreviations]{glossaries-extra}

you can simply do:

\usepackage[abbreviations,nonumberlist]{glossaries-extra}

It’s better not to load the glossaries package first. Leave glossaries-extra to load it, where possible, to allow for a smoother integration between the two packages.

After glossaries-extra has been loaded, some of the glossaries-extra package options may be changed with:

\glossariesextrasetup{(options)}

where (options) are the same as the relevant package option.

Certain options can only be supplied as package options since the settings need to be known while glossaries-extra is loading.

To change the base glossaries package’s options (that may be changed after the package
2. Package Options

has loaded), continue to use:

\setupglossaries\{⟨options⟩\}

but don’t use any of the options listed here in that command.

2.1. Glossary Lists

nomissingglstext=(boolean)  
default: true; initial: false

If true, this will suppress the warning written to the transcript and the warning text that will appear in the document if the external glossary files haven’t been generated due to an incomplete document build. However, it’s probably simpler just to fix whatever has caused the failure to build the external file or files.

abbreviations

This option has no value and can’t be cancelled. If used, it will automatically create a new glossary with the label abbreviations and redefines \glsxtrabbrvtype to this label. (The file extensions are glg-abr, gls-abr and glo-abr.) In addition, this option defines a shortcut command:

\printabbreviations[⟨options⟩]

which is equivalent to:

\printglossary[type=glxtrabbrvtype,⟨options⟩]

If glossaries-extra-bib2gls is also loaded then this option will additionally provide \printunsrtabbreviations which uses \printunsrtglossary instead.

The title of the new glossary is given by

\abbreviationsname  
initial: Abbreviations

If this command is already defined, it’s left unchanged. Otherwise it’s defined to “Abbreviations” if babel hasn’t been loaded or \acronymname if babel has been loaded. However, if you’re using babel it’s likely you will need to change this. (See §15 for further details.)
If you don’t use the `abbreviations` package option, the \abbreviationsname command won’t be defined (unless it’s defined by an included language file).

If the `abbreviations` option is used and the `acronym` option provided by the `glossaries` package hasn’t been used, then \acronymtype will be set to \glsxtrabbrvtype so that acronyms defined with \newacronym can be added to the list of abbreviations. If you want acronyms in the main glossary and other abbreviations in the `abbreviations` glossary then you will need to redefine \acronymtype to `main`:

\renewcommand*{\acronymtype}{main}

Note that there are no analogous options to the `glossaries` package’s `acronymlists` option (or associated commands) as the abbreviation mechanism is handled differently with `glossaries-extra`.

`symbols`
This is passed to the base `glossaries` package, but `glossaries-extra` will additionally define:

\glsxtrnewsymbol[⟨key=value list⟩]{⟨entry-label⟩}{⟨sym⟩}

which is equivalent to:

\newglossaryentry{⟨entry-label⟩}{name={⟨symbol⟩}, sort={⟨entry-label⟩}, type={symbols}, category={symbol}, ⟨options⟩}

Note that the `sort` key is set to the `⟨entry-label⟩` not the `⟨symbol⟩` as the symbol will likely contain commands. If this isn’t appropriate, you can override it by using the `sort` key in the optional argument.

This option also sets the `regular` attribute to `true` for the `symbol` category and provides the category post-description hook:

\glsxtrpostdescsymbol initial: empty

If `glossaries-extra-bib2gls` is also loaded then this option will additionally provide \print-
unsrtsymbols which uses \printunsrtglossary.

\begin{itemize}
\item \textbf{numbers}
\end{itemize}

This is passed to the base glossaries package but glossaries-extra will additionally define:

\begin{verbatim}
\glsxtrnewnumber[(key=value list)]{(entry-label)}{(num)}
\end{verbatim}

which is equivalent to:

\begin{verbatim}
\newglossaryentry{(entry-label)}{name={⟨number⟩},
    sort={⟨entry-label⟩},type={numbers},category={number},
    (options)}
\end{verbatim}

Note that the \texttt{sort} key is set to the \texttt{⟨entry-label⟩}. If this isn’t appropriate, you can override it by using the \texttt{sort} key in the optional argument.

This option also sets the \texttt{regular} attribute to \texttt{true} for the \texttt{number} category and provides the category post-description hook:

\begin{verbatim}
\glsxtrpostdescnumber initial: empty
\end{verbatim}

If glossaries-extra-bib2gls is also loaded then this option will additionally provide \printunsrtnumbers which uses \printunsrtglossary.

\textbf{acronyms}

This is passed to the base glossaries package (which defines \printacronyms and creates a new glossary with the label acronym) but if glossaries-extra-bib2gls is loaded then this option will additionally provide \printunsrtacronyms which uses \printunsrtglossary.

As with the base glossaries package, this option redefines \texttt{\acronymtype} to \texttt{acronym}. Note that this option doesn’t change \texttt{\glsxtrabbrvtype}.

\begin{verbatim}
\acrofinal{(boolean)} default: true; initial: false
\end{verbatim}

If \texttt{acronym=true}, this behaves like \texttt{acronyms}. Note that \texttt{acronym=false} won’t work if the base glossaries package was loaded before glossaries-extra.
2. Package Options

This is passed to the base glossaries package but if glossaries-extra-bib2gls is loaded then this option will additionally provide \printunsrtindex which uses \printunsrtglossary.

The base package index option also defines:

\newterm{\{key=value list\}\{entry-label\}}

This definition is modified by glossaries-extra to additionally set the category to index and sets the description to discard the post-description hook (\nopostdesc) but retain \glsxtrpostdescription so that the category post-description hook can still be applied.

This option also sets the regular attribute to true for the index category and defines an associated category post-description hook:

\glsxtrpostdescindex initial: empty

2.2. Glossary Style Options

\nopostdot=\{boolean\} default: true; initial: true

This option is provided by glossaries where it simply alters a corresponding conditional that’s used inside \glspostdescription to determine whether or not to insert a full stop.

The postpunc option (see below) redefines \glspostdescription, so the nopostdot option is modified by glossaries-extra to reset the hook back to its original definition to counteract any use of the postpunc option.

This option will have no effect if the glossary style doesn’t include \glspostdescription. (Use stylemods to ensure that all the predefined styles that show the description have this hook added.)

If you are using bib2gls, you may prefer to use the post-description-dot resource option.

postdot

This is a shortcut for nopostdot=false.

postpunc=\{value\}
This option redefines `\glspostdescription` to display the required punctuation. Note that this means the hook will no longer check for the `nopostdot` conditional.

This option will have no effect if the glossary style doesn’t include `\glspostdescription`. (Use `stylemods` to ensure that all the predefined styles that show the description have this hook added.)

The `postpunc` value may either be the required punctuation or one of the following keywords:

- `postpunc=dot`
  This redefines `\glspostdescription` to use a full stop but also adjusts the space factor. This isn’t exactly the same as `nopostdot=false` since it removes the conditional from `\glspostdescription`. If you are using `bib2gls`, you may prefer to use the `post-description-dot` resource option.

- `postpunc=comma`
  This redefines `\glspostdescription` to a comma.

- `postpunc=none`
  This redefines `\glspostdescription` to do nothing. This isn’t exactly the same as `nopostdot=true` since it removes the conditional from `\glspostdescription`.

- `stylemods=<value>`
  `default: default`
  Loads the `glossaries-extra-stylemods` package (see §8.6.5), which patches the styles provided with the base `glossaries` package so that they all use `\glspostdescription`. Extra hooks are also provided to make them easier to customize. The value may be one of the following:

- `stylemods=all`
  Loads all styles that are provided by both `glossaries` and `glossaries-extra`. 

- `stylemods=default`
2. Package Options

Patches all the predefined styles that have been loaded, without loading any extra styles. This will typically be all the styles that are usually loaded by glossaries (for example, list and long). Package options such as nolist will alter which styles are loaded. In the case of nostyles, no styles will be loaded, so none of them will be patched.

It’s pointless using both stylemods=default and nostyles. Any glossary style packages that are subsequently loaded won’t be patched.

\texttt{stylemods=\langle list \rangle}

For each element \texttt{\langle tag \rangle} in \texttt{\langle list \rangle}, the corresponding package glossary-\texttt{\langle tag \rangle} will be loaded. You can use this in combination with nostyles to only load the particular style package or packages that you require (without loading the full set of defaults). For example,

\begin{verbatim}
\usepackage[nostyles,stylemods={bookindex,longextra}, style=bookindex]{glossaries-extra}
\end{verbatim}

This prevents the base glossaries package from loading the default set of styles, but loads glossaries-extra-stylemods, glossary-bookindex and glossary-longextra, and then sets the default style to bookindex.

2.3. Loading Other Packages

Some options listed in other sections, such as the stylemods and record options, also load supplementary packages.

\texttt{prefix}

Loads the glossaries-prefix package (if not already loaded).

\texttt{accsupp}

Loads the glossaries-accsupp package (if not already loaded). This option can only be used as a package option (not in \texttt{\glossariesextrasetup}) as glossaries-extra needs to know whether or not to provide accessibility support while it’s loading.
The glossaries-accsupp package is still experimental and so accessibility features are liable to change.

If you want to define styles that can interface with the accessibility support provided by glossaries-accsupp use the \glsaccess\langle xxx\rangle type of commands instead of \glsentry\langle xxx\rangle (for example, \glsaccesstext instead of \glsentrytext). If glossaries-accsupp hasn’t been loaded those commands are equivalent (for example, \glsaccesstext just does \glsentrytext) but if it has been loaded, then the \glsaccess\langle xxx\rangle commands will add the accessibility information. See §9 for further details.

2.4. Entry Definitions, References and Indexing

\texttt{\texttt{undefaction}=(\texttt{value}) \hspace{1cm} \textit{initial: error}}

This indicates what to do if an undefined glossary entry is referenced.

Undefined entries can’t be picked up by any commands that iterate over a glossary list. This includes \forglsentries and \glsaddall.

\texttt{\texttt{undefaction}=\texttt{error}}

Produces an error message for undefined glossary entries.

\texttt{\texttt{undefaction}=\texttt{warn}}

Only produces a warning message for undefined glossary entries. The place where the entry has been referenced will be marked with ?? (as with undefined labels or citations). The unknown marker is produced with:

\texttt{\glsxtrundeftag \hspace{1cm} \textit{initial: ??}}

This defaults to two question marks.

Note that \ifglsused will only display ?? in the document text with \texttt{undefaction=warn} if the entry hasn’t been defined, as the underlying boolean variable doesn’t exist and so is neither true nor false. (There will also be a warning in the transcript.) You may prefer to use \GlsXtrIfUnusedOrUndefined instead. See §5.10 for further details.
2. Package Options

If you want to write a custom command that needs to generate a warning or error for an undefined reference, you can use:

\glsxtrundefaction{⟨message⟩}{⟨additional help⟩}

This will produce the unknown marker if used within the document environment. Depending on the undefaction, \glsxtrundefaction will either create an error with the given ⟨message⟩ and ⟨additional help⟩ or will create a warning with the given ⟨message⟩.

\docdef=(value) \hspace{1cm} default: true; initial: false

This setting governs where \newglossaryentry can be used (preamble-only or anywhere before the first glossary or anywhere within the document).

Commands like \newabbreviation and \glsxtrnewsymbol that internally use \newglossaryentry are also governed by this option. Other commands, such as \longnewglossaryentry are always preamble-only.

With just the base glossaries package, \newglossaryentry is allowed in the document environment as long as you haven’t used \makenoidxglossaries. There are, however, problems that can occur when entries are defined within the document environment (see the glossaries documentation for further details). To encourage preamble-only use, the glossaries-extra package prohibits the use of \newglossaryentry within the document environment by default, but if you really want this you can use this package option to allow it.

Note that in the case of bib2gls, all entry data is originally defined in bib files. The entry definitions (using commands like \longnewglossaryentry and \newabbreviation) are written to the glstex files that are input in the preamble.

\docdef=false

Prohibits the use of \newglossaryentry within the document environment. All entries must be defined in the preamble.

\docdef=true

Permits the use of \newglossaryentry in the document environment provided \makenoidxglossaries hasn’t been used (as per the base glossaries package). This will create a temporary glsdefs file that contains the entry definitions so that they can be available on the next \LaTeX run at the beginning of the document to allow any glossaries in the front matter to display correctly.
If all your glossaries occur at the end of the document, consider using \texttt{docdef=restricted} instead.

\begin{center}
\texttt{docdef=restricted}
\end{center}

Permits the use of \texttt{\newglossaryentry} in the document environment provided the entry definitions all occur before the first glossary is displayed.

This avoids the need for the \texttt{glsdefs} file. You will still need to take care about any changes made to the category code of characters that are required by the \texttt{\langle key\rangle=\langle value\rangle} mechanism (that is, the comma and equal sign) and any\texttt{makeindex} or \texttt{xindy} special character that occurs in the \texttt{sort} key or label. If any of those characters are made active in the document (for example, through \texttt{babel} shortcuts), then it can cause problems with the entry definition.

This option will allow \texttt{\newglossaryentry} to be used in the document with \texttt{\makenoindexglossaries}, but note that \texttt{\longnewglossaryentry} remains a preamble-only command.

With this option, if an entry appears in the glossary before it has been defined, an error will occur (or a warning if the \texttt{undefaction=warn} option is used). If you edit your document and either remove an entry or change its label, you may need to delete the document’s temporary files (such as the aux and gls files).

\begin{center}
\texttt{docdef=atom}
\end{center}

This option behaves like \texttt{docdef=restricted} but creates the \texttt{glsdefs} file for atom’s autocomplete support. This file isn’t input by \texttt{glossaries-extra} and so associated problems with the use of this file are avoided, but it allows the autocomplete support to find the labels in the file.

A bug fix in \texttt{glossaries v4.47} has changed the format of the \texttt{glsdefs} file slightly.

As with \texttt{docdef=restricted}, entries may be defined in the preamble or anywhere in the document, but they may only be referenced after they have been defined. Entries must be defined before the associated glossary is displayed.

If you need a list of all entry labels for the use of an editor or helper script you may also want to consider the package options \texttt{writeglslabels} and \texttt{writeglslabelnames} provided by the base \texttt{glossaries} package. Note that with these options and with \texttt{docdef=atom}, only the entry labels that are visible to \texttt{EPiX} can be saved. So if you are using \texttt{bib2gls} you will only get the labels of the entries that have already been selected by \texttt{bib2gls}. The \texttt{bib} files can be found by parsing the aux file for \texttt{\glsxtr@resource} (listed in the \texttt{src} option or \texttt{\jobname.bib} if \texttt{src} is missing).

\begin{center}
\texttt{shortcuts=\{\langle value\rangle\}} \hspace{1cm} \textit{initial: none}
\end{center}
2. Package Options

Unlike the base glossaries package option of the same name, this option isn’t boolean but has multiple values.

Multiple invocations of the `shortcuts` option within the same option list will override each other. Since these options define commands, the action can’t be undone with a later \glossariesextrasetup.

- **shortcuts=ac**

Set the shortcut commands provided by the base glossaries package for acronyms (such as \ac) but use the glossaries-extra abbreviation commands, such as \glsxtrshort and \glsxtrlong, instead of the analogous base commands, such as \acrshort and \acrlong. See §4.3.2 for further details.

- **shortcuts=abbreviations**

Sets the abbreviation shortcuts (see §4.3.2). This setting doesn’t switch on the acronym shortcuts provided by the base glossaries package.

- **shortcuts=abbr**

Alias: abbreviations

Synonym for `shortcuts=abbreviations`.

- **shortcuts=other**

Implements the other (non-abbreviation) shortcut commands:

\newentry{⟨entry-label⟩}{⟨options⟩}

A synonym for \newglossaryentry.

\newsym[(key=value list)]{⟨entry-label⟩}{⟨sym⟩}

A synonym for \glsxtrnewsymbol (provided that the symbols package option is also used).
2. Package Options

\newnum{⟨key=value list⟩}{⟨entry-label⟩}{⟨num⟩}

A synonym for \glstextrnewnumber (provided that the numbers package option is also used).

**shortcuts=acother**

Implements shortcuts=ac and shortcuts=other.

**shortcuts=abother**

Implements shortcuts=abbreviations and shortcuts=other.

**shortcuts=all**

Implements shortcuts=ac, shortcuts=abbreviations and shortcuts=other.

**shortcuts=acronyms**

Sets the shortcuts provided by the base glossaries package for acronyms (such as \ac). See the glossaries package documentation for further details.

Note that the short and long forms (\acs and \acl) don’t use \glstextrshort and \glstextrlong but use the original \acrshort and \acrlong, which aren’t compatible with the glossaries-extra abbreviation mechanism. The better option is to use shortcuts=ac.

Don’t use shortcuts=acronyms unless you have reverted \newacronym back to the base glossaries package’s acronym mechanism. See §4.6 for further details.

**shortcuts=acro**

alias: acronyms

Synonym for shortcuts=acronyms.

**shortcuts=true**

alias: all

This setting is provided by the base glossaries package. With glossaries-extra it’s equivalent
2. Package Options

to `shortcuts=all`.

This setting is provided by the base glossaries package. With `glossaries-extra` it’s equivalent to `shortcuts=none`.

`shortcuts=false`  
`alias: all`

This is a boolean option that governs whether or not to automatically index any cross-referenced entries that haven’t been marked as used at the end of the document. These are entries that are identified in one of the cross-referencing fields (`see` and `seealso`) of another used entry as opposed to entries that have the cross-referencing fields set.

Since entries with the `alias` key are intended as synonyms for another term, the target is expected to be indexed so entries with the `alias` key set aren’t affected by this option.

For example:

```latex
\newglossaryentry{courgette}{name={courgette},
    description={small vegetable marrow}}
\newglossaryentry{marrow}{name={marrow},
    description={long gourd with green skin},
    seealso={courgette}}
```

Suppose that “marrow” is indexed (so that it appears in the glossary with the cross-reference to “courgette”) but if courgette isn’t indexed anywhere in the document (using commands like `\gls` or `\glsadd`) then there will be a broken cross-reference in the marrow location list pointing to courgette, which doesn’t appear in the glossary. With `indexcrossrefs=true`, the courgette entry will be indexed at the end of the document using `\glsadd` with format `\glsxtrunusedformat`, which corresponds to the command `\glsxtrunusedformat`.

Note that this special format `\glsxtrunusedformat` simply does `\unskip` and ignores its argument, which creates a blank location. If any of the cross-referenced entries have been indexed but haven’t been marked as used (for example, with `\glsadd`) then this will cause a spurious comma in the location list. This is a limitation of the way that makeindex and xindy work as they are general purpose indexing applications which require locations. If you have entries with cross-references, you may want to consider switching to `bib2gls` instead.

Note that `bib2gls` can automatically find dependent entries when it parses the bib source file, so the `record` option automatically implements `indexcrossrefs=false`.

20
2. Package Options

This function is implemented by code added to the end document hook that determines whether or not to use the command \glsxtraddallcrossrefs. This command iterates over all entries in all glossaries, which adds to the overall document build time, especially if you have defined a large number of entries, so this defaults to indexcrossrefs=false, but it will be automatically switched on if you use the see or seealso keys in any entries. See also §5.9.

<table>
<thead>
<tr>
<th>indexcrossrefs=true</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enables this setting.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>indexcrossrefs=false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disables this setting even if the see or seealso key is present in any entries.</td>
</tr>
</tbody>
</table>

| autoseeindex={boolean} | default: true; initial: true |
|-----------------------|
| This is a boolean option that governs whether or not the see and seealso keys should automatically index the cross-reference when an entry is defined (see §5.9). |

With the base glossaries package, the see key was provided as a shortcut for \glssee. For example:

\newglossaryentry{courgette}{name={courgette},
  description={small vegetable marrow}}
\newglossaryentry{zucchini}{name={zucchini},
  description={},
  see={courgette}}

is equivalent to:

\newglossaryentry{courgette}{name={courgette},
  description={small vegetable marrow}}
\newglossaryentry{zucchini}{name={zucchini},
  description={}}
\glssee{zucchini}{courgette}

This was designed for documents where only entries that are actually used in the document are defined and ensures that the cross-reference is included in the glossary, even though it may not be referenced anywhere in the document. However, it becomes problematic if neither entry is required in the document.
2. Package Options

The glossaries-extra package modifies the action of the see key so that it also saves the value and will only perform the automated \glssee if autoseeindex=true. Similarly for the seealso key.

\begin{quote}
Note that the record option automatically implements autoseeindex=false as the corresponding action can be implemented with bib2gls’s selection option.
\end{quote}

\begin{quote}
\textbf{autoseeindex=true}
\end{quote}

Enables automatic indexing using \glssee for the see key (as per the base glossaries package) and \glsxtrindexseealso for the seealso key.

For example, if an entry is defined as

\begin{verbatim}
\newglossaryentry{foo}{name={foo},
description={},see={bar,baz}}
\end{verbatim}

then, with autoseeindex=true and the default indexcrossrefs setting, this is equivalent to

\begin{verbatim}
\newglossaryentry{foo}{name={foo},description={}}
\glssee{foo}{bar,baz}
\glossariesextrasetup{indexcrossrefs=true}
\GlsXtrSetField{foo}{see}{bar,baz}
\end{verbatim}

\begin{quote}
\textbf{autoseeindex=false}
\end{quote}

The value of the see and seealso keys will be stored in their corresponding fields (and can be accessed using commands like \glsxtrusesee and \glsxtruseseealso) but the cross-reference won’t be automatically indexed.

\begin{quote}
Note that indexcrossrefs isn’t automatically implemented by the presence of the see key when autoseeindex is false.
\end{quote}

For example, if an entry is defined as
2. Package Options

\newglossaryentry{foo}{name={foo},
description={},see={bar,baz}}

then, with \texttt{autoseeindex=false} and the default \texttt{indexcrossrefs} setting, this is equivalent to

\newglossaryentry{foo}{name={foo},
description={}}
\GlsXtrSetField{foo}{see}{bar,baz}

It’s therefore possible with this option to remove the cross-references from the location lists and set their position within the glossary style.

Another method of preventing the automatic indexing is to define the entries before the external indexing files have been opened with \texttt{\makeglossaries}. Since the appropriate file isn’t open, the information can’t be written to it. This will need the package option \texttt{seenoindex=ignore} to prevent an error occurring.

\texttt{record=(value)} \hspace{1cm} \texttt{default: only; initial: off}

This setting indicates whether or not bib2gls is required.

This option can only be set in the preamble and can’t be used after \texttt{\GlsXtrLoadResources} or \texttt{\glsxtrresourcefile}.

With the recording setting on (\texttt{record=only}, \texttt{record=nameref} or \texttt{record=hybrid}), any of the commands that would typically index the entry (such as \texttt{\gls}, \texttt{\glstext} or \texttt{\glsadd}) will add a record to the aux file. bib2gls can then read this information to find out which entries have been used. (Remember that commands like \texttt{\glsentryname} don’t index, so any use of these commands won’t add a corresponding record.) See §11 for further details.

The hybrid method additionally performs the standard indexing action that’s required for \texttt{makeindex} or \texttt{xindy} to work, but this can’t be done until bib2gls has created the glstex files that provide the entry definitions. In general, it’s best to avoid the hybrid method.

\texttt{record=off}

Indexing is performed as per the base glossaries package using either \texttt{\makeglossaries} or


2. Package Options

\makenoidxglossaries. This setting implements \texttt{undefaction=error}.

\begin{Verbatim}
record=only
\end{Verbatim}

Indexing (or recording) is performed by adding \texttt{bib2gls} records in the aux file. Neither \texttt{\makeglossaries} nor \makenoidxglossaries is permitted. Use \texttt{\GlsXtrLoadResources} (or \texttt{\glsxtrresourcefile}) to set up \texttt{bib2gls} resource options. Glossaries should be displayed with the “unsrt” family of commands, such as \texttt{\printunsrtglossary}.

This setting implements \texttt{undefaction=warn}, \texttt{autoseeindex=false}, \texttt{indexcrossrefs=false sort=none}, and automatically loads the supplementary glossaries-extra-\texttt{bib2gls} package. (There should be no need to explicitly load \texttt{glossaries-extra-\texttt{bib2gls}}.)

This option also defines the \texttt{location} and \texttt{group} keys that are set by \texttt{bib2gls} to provide the location list and group information required by the “unsrt” family of commands.

The document build process is (assuming the file is called \texttt{myDoc.tex}):

\begin{Verbatim}
pdflatex myDoc
bib2gls myDoc
pdflatex myDoc
\end{Verbatim}

If you want letter groups you will need the \texttt{--group} or \texttt{-g} switch when invoking \texttt{bib2gls}:

\begin{Verbatim}
pdflatex myDoc
bib2gls -g myDoc
pdflatex myDoc
\end{Verbatim}

Note that this setting will prevent the \texttt{see} from automatically implementing \texttt{\glssee}. (\texttt{bib2gls} deals with the \texttt{see} field.) You may explicitly use \texttt{\glssee} in the document, but \texttt{bib2gls} will ignore the cross-reference if the \texttt{see} field was already set for that entry.

\begin{Verbatim}
record=nameref
\end{Verbatim}

As \texttt{record=only} but uses \texttt{nameref} records, which include the current label information given by \texttt{\@currentlabel} and \texttt{\@currentHref}. This means that the title can be included in the entry locations, if available. This setting also supports location hypertargets that don’t follow a simple \texttt{⟨h-prefix⟩⟨the-counter⟩} format, which can’t be used with other indexing options.

See §11.5.6 for further details of this option.
2. Package Options

This option requires hyperref, otherwise it will fall back on the usual location records.

Note that \@currentHref is always globally updated whenever \refstepcounter is used, but \@currentlabel isn’t. This can cause some undesired side-effects with some settings. Remember also that the indexcounter option increments the associated counter every time an entry is indexed, which affects this option. If the location counter is the default page, only the location number is shown.

\texttt{record=alsoindex} \hfill \textit{alias: hybrid}

Deprecated synonym of record=hybrid.

\texttt{record=hybrid}

This is a hybrid setting that uses bib2gls to fetch entry information from bib files, but uses makeindex or xindy to create the glossary files (which are input with \texttt{\printglossary}). Note that this requires a slower and more complicated build process (see below).

This hybrid approach is provided for the rare instances where an existing xindy rule or module is too complicated to convert to a bib2gls rule but the entries need to be fetched from a bib file. There’s no benefit in using this option with makeindex.

This setting does not load glossaries-extra-bib2gls, as bib2gls is only being used to fetch the entry definitions.

Since it’s redundant to make bib2gls also sort and collate locations (in addition to xindy performing these tasks), use the resource options sort=none and save ~locations=false for a faster build. Many of the other resource options are likely to be irrelevant.

This setting must be used with \texttt{\makedglossaries} but not with its optional argument. Each glossary should be displayed using \texttt{\printglossary} (or \texttt{\printglossaries} for all of them).

This setting should not be used with \texttt{\makenoidxglossaries}.

You may need to change the transcript file used by bib2gls to avoid a clash with xindy’s transcript file. This can be done with bib2gls’s --log-file or -t option.

The document build process is (assuming the file is called myDoc.tex):
2. Package Options

```
pdflatex myDoc
bib2gls myDoc
pdflatex myDoc
makeglossaries myDoc
pdflatex myDoc
```

Note that, in this case, it's redundant to call bib2gls with the `--group` or `-g` switch as xindy will insert the group heading information into the corresponding glossary file.

If you want bib2gls to form the letter groups then this hybrid method is inappropriate.

```
bibglsaux={basename} initial: empty
requires bib2gls v3.0+
```

Alternatively, this setting can be implemented with:

```
\glsxtrsetbibglsaux{⟨basename⟩}
```

This option should only be used once. If used again no new file will be created. If the `⟨basename⟩` value is empty, records will be written to the normal `aux` file.

A document containing many records can result in a large `aux` file with information that's only relevant to bib2gls. This option will create a new file called `⟨basename⟩.aux` that will be used to store the records. The file will be skipped by LaTeX but will be picked up by bib2gls v3.0+ when it inputs the main `aux` file. Note that this creates an extra write register.

You should still supply the main `aux` file when you run bib2gls as `⟨basename⟩.aux` will only contain the records and not the other information that bib2gls requires (such as the resource options).

```
equations={boolean} default: true; initial: false
```

This setting will cause the default location counter to automatically switch to equation when inside a numbered equation environment, such as `equation` or `align`. The counter can be
explicitly overridden with the `counter \glslink` option.

---

### floats

- **Default:** true; **Initial:** false

This setting will cause the default location counter to automatically switch to the corresponding counter when inside a floating environment, such as `figure` or `table`. The counter can be explicitly overridden with the `counter \glslink` option.

Remember that within floats it’s the `\caption` command that actually uses `\refstepcounter`, so indexing before the caption will result in the wrong reference. The commands for use in captions and sections, such as `\glssfmttext` and `\glssfmtshort`, don’t index. (See §5.3). You may want to consider using `\glsadd` after the caption (not before). For example:

```latex
\begin{figure}[htbp]
  \centering
  \includegraphics{example-image}
  \caption{Sample \glssfmttext{foobar} figure}
  \glsadd{foobar}
\end{figure}
```

### indexcounter

This option defines the indexing counter:

```latex
\texttt{wrglossary}
```

which is incremented every time an entry is indexed. This option automatically implements `counter=wrglossary`. This means that each location will link to the relevant part of the page where the indexing occurred (instead of to the top of the page). See the `bib2gls` documentation for the `save-index-counter` resource option for more details.

---

This option works by incrementing `wrglossary` with `\refstepcounter` and adding `\label`. This can cause a problem if the indexing occurs in an equation environment as `amsmath` forbids multiple occurrences of `\label` (resulting in the "Multiple \label’s" error). It’s best to change the counter to `page` or `equation` when in maths mode with this option. For example:
2. Package Options

\renewcommand{\glslinkpresetkeys}{%\ifmmode \setupglslink{counter=page}\fi}\renewcommand{\glsaddpresetkeys}{%\ifmmode \setupglsadd{counter=page}\fi}

2.5. Debugging

\textbf{debug=(value)} \textit{default: true; initial: false}

Enables debugging information for draft documents. This option is defined by the base glossaries package, but is extended by glossaries-extra to provide additional settings. If no value is provided, true is assumed. The following values are available:

\begin{itemize}
  \item \textbf{debug=false}
  \begin{itemize}
    \item This setting is provided by the glossaries package and is the default. This switches off all debugging commands.
  \end{itemize}
  \item \textbf{debug=true}
  \begin{itemize}
    \item This setting is provided by the glossaries package and switches on logging information if an entry is indexed before the relevant indexing files have been opened (only applicable with makeindex and xindy). This option is extended by glossaries-extra to also display the label of unknown entries before the ?? marker.
  \end{itemize}
\end{itemize}

\begin{verbatim}
\documentclass{article} \usepackage[debug]{glossaries-extra} \begin{document} \gls{example} \end{document}
\end{verbatim}

This uses \texttt{\glsshowtargetfonttext} for the annotation, which is provided by glossaries.

\textbf{debug=showaccsupp}
2. Package Options

Provided by glossaries, this setting shows accessibility information (glossaries-accsupp).

```
debug=all
```

Implements all debugging options.

```
debug=showwrgloss
```

This setting is only available with glossaries-extra. This implements debug=true and shows a marker (·) just before the write operation is performed by indexing commands. With record=hybrid there will be two marks: one for the write operation to the aux file and one for the associated glossary file used by makeindex/xindy. The marker is produced with the command:

```
\glsxtrwrglossmark
```

If the indexcounter option has been used, this setting will also mark where the wrglossary counter has been incremented. The marker is produced with the command:

```
\glsxtrwrglosscountermark{⟨number⟩}
```

This marker is also inserted before the location in the definition of \glsxtrwrglossarylocfmt.

```
debug=showtargets
```

This setting is provided by glossaries and displays the hyperlink target names whenever any glossary-related commands create a hyperlink or hypertarget (for example, \gls, \gls-\target or \glshypernumber). The default is to use marginal notes in \TeX’s “outer” mode and inline annotations for “inner” or maths modes. This uses \glsshowtargetinner for inner and maths annotations and \glsshowtargetouter for the outer annotation.

If there are many targets within a single paragraph this can lead to “too many floats”, so glossaries-extra provides a new package option showtargets that can be used to easily switch to inline annotations for outer mode (rather than having to redefine \glsshowtargetouter).

```
showtargets=⟨value⟩
```

29
Automatically implements `debug=showtargets` and adjusts the annotations according to the ⟨value⟩. The glossaries-extra package provides supplementary commands to support this option.

\glsxtrshowtargetouter{⟨target-name⟩}

Formats annotations in outer mode. This is initially \gls@showtargetouter to match `debug=showtargets`.

\glsxtrshowtargetinner{⟨target-name⟩}

Formats annotations in inner mode. This is initially \gls@showtargetinner to match `debug=showtargets`.

\glsshowtargetinnersymleft{name}

Shows the left annotation and marker. This uses the left symbol marker:

\glsxtrshowtargetsymbolleft

\glsshowtargetinnersymright{name}

Shows the right marker and annotation. This uses the right symbol marker:

\glsxtrshowtargetsymbolright

`showtargets=left`

A marker is placed to the left of the link/target and a marginal note is used in outer mode.

`showtargets=right`
A marker is placed to the right of the link/target and a marginal note is used in outer mode.

```latex
showtargets=innerleft
```

A marker and annotation are placed to the left of the link/target in all modes.

```latex
showtargets=innerright
```

A marker and annotation are placed to the right of the link/target in all modes.

```latex
showtargets=annoteleft
```

Markers are placed on either side of the link/target with the annotation on the left in all modes.

```latex
showtargets=annoteright
```

Markers are placed on either side of the link/target with the annotation on the right in all modes.
3. Defining Entries

The base glossaries package provides commands, such as \newglossaryentry, to define entries. The glossaries-extra package provides some additional commands, described in §3.1. For abbreviations, see §4. If you use bib2gls, it will write command definitions within the glstex file. See the bib2gls user manual for further information about those commands.

The glossaries user manual warns against using commands such as \gls within field values. However, if you really need this, the glossaries-extra package provides \glsxtrp (see §5.4). Alternatively, you may want to consider multi (compound) entries instead (see §7).

3.1. Command Definitions

\longnewglossaryentry{⟨entry-label⟩}{⟨key=value list⟩}{⟨description⟩}

This command is provided by the base glossaries package to cater for entries with descriptions that contain paragraph breaks. (The ⟨key⟩=⟨value⟩ interface doesn’t support paragraph breaks in the value.) The base package only provides an unstarred version of this command, which automatically inserts \leavevmode\unskip\nopostdesc at the end of the description. The glossaries-extra package replaces this with a single command:

\glsxtrpostlongdescription

which has the same effect, but can be redefined if required.

The glossaries-extra package provides a starred form:

\longnewglossaryentry*{⟨entry-label⟩}{⟨key=value list⟩}{⟨description⟩}

This doesn’t insert the hook at the end of the description.

For a general purpose post-description hook, see §8.6.2.

Additionally, the symbols package option provides \glsxtrnewsymbol, and the numbers package option provides \glsxtrnewnumber. See §2.1 for further details.
3. Defining Entries

3.2. Glossary Entry Keys

In addition to the glossary entry keys provided by the base glossaries package (summarised in §II) the glossaries-extra package provides:

- **category**={*category-label*}  
  *initial: general*

  Assigns the category label. This should not contain any special or active characters as it’s used to form command names. See §10 for further details.

- **seealso**={*xr-list*}  

  This key is analogous to the see key but the tag is always given by \seealso{name}. The value (*xr-list*) should be a comma-separated list of entry labels. As with the see key, this key automatically indexes the cross-reference by default. The cross-reference will be displayed in the location list using \glsxtruseseealsoformat (see §5.9). Use autoseeindex=false to prevent the automatic indexing. (With bib2gls, adjust the selection criteria.)

  With just the base glossaries package, the see key simply performs this automated indexing. With glossaries-extra the value is also saved. Similarly with the seealso key. The value isn’t saved with explicit use of \glsxtrindexseealso or \glssee.

- **alias**={*xr-label*}  

  This is similar to the see key but the value can only be a single entry label. In addition to automatically indexing the cross-reference, this command will cause the entry with this key to have hyperlinks go to the aliased entry when referenced with commands like \gls. Whenever the entry is indexed with commands like \gls, the indexing will be performed on the target entry (the alias value). See §5.9 for further details.

  Any entry that has a see, seealso or alias key set will be added to the glossary by default when using makeindex or xindy. If you don’t want this behaviour, use the autoseeindex=false package option and implement a post-description hook to insert the cross-reference. Alternatively, consider switching to bib2gls.

  If you use bib2gls (see §11) then most of the glossary entry keys can be used as analogous fields in the bib file. For example, instead of writing the following code in your tex file:

\begin{verbatim}
```
3. Defining Entries

\newglossaryentry{duck}{name={duck},
    description={a waterbird with webbed feet}}
\newabbreviation{svm}{SVM}{support vector machine}

You would write the following in a bib file:

@entry{duck,
    name={duck},
    description={a waterbird with webbed feet}}
@abbreviation{svm,
    short={SVM},
    long={support vector machine},
}

There are, however, some keys that are considered internal fields by bib2gls, in that they
are defined as keys by glossaries-extra and may be assigned in the glstex file that’s input
by \GlsXtrLoadResources, but they should not be used in the bib files.

For example, the sort key (which is recommended with xindy where the name contains
symbols) should not be used in the bib file. Instead, use the sort-field resource option or
the system of sort fallbacks to choose the most appropriate field to obtain the sort value (see
Gallery: Sorting\textsuperscript{1}). The group and location keys are also considered internal fields and are
only applicable with the “unsrt” family of commands.

The group and location keys are defined by the record=only and record=nameref
options and are only applicable with the “unsrt” family of commands.

\texttt{\texttt{group=\{(group-label)\}}}

This key is used by bib2gls within the glstex file to set the group label. This label is typi-
cally a by-product of the sorting method (see §8.6.4). If it is explicitly set without reference
to the order it can result in fragmented groups, see Gallery: Logical Glossary Divisions (type
vs group vs parent).\textsuperscript{2} The group title can be set with \texttt{\glsxtrsetgrouptitle}. You will
need to invoke bib2gls with the --group (or -g) switch to ensure that this key is set, when
required.

\textsuperscript{1}dickimaw-books.com/gallery/index.php?label=bib2gls-sorting
\textsuperscript{2}dickimaw-books.com/gallery/index.php?label=logicaldivisions

34
3. Defining Entries

Letter groups are a consequence of sorting, not the other way around.

\( \text{location} = \{ \langle \text{location-list} \rangle \} \)

Used by \texttt{bib2gls} to store the formatted location list. The unformatted internal location list is stored in the \texttt{loclist} field, as with \texttt{printnoidxglossary}.

With the "unsrt" family of commands, if the \texttt{location} field isn’t set, then it will try the \texttt{loclist} field instead, using the same method as \texttt{printnoidxglossary} to display the locations. If you don’t want locations with \texttt{bib2gls}, either use \texttt{nonumberlist} or use the \texttt{save-locations=false} resource option.

The base \texttt{glossaries} package provides \texttt{\glsaddkey} and \texttt{\glsaddstoragekey} to allow custom keys to be defined. The \texttt{glossaries-extra} package additionally provides:

\begin{itemize}
  \item \texttt{\glsxtrprovidestoragekey{\langle key \rangle}{\langle default value \rangle}{\langle no link cs \rangle}} modifier: *
  \begin{itemize}
    \item This is like \texttt{\glsaddstoragekey} but does nothing if the key has already been defined. As with \texttt{\glsaddstoragekey}, the starred version switches on field expansion for the given key (provided that it hasn’t already been defined).
  \end{itemize}
  \item \texttt{\glsxtrifkeydefined{\langle key \rangle}{\langle true \rangle}{\langle false \rangle}}
  \begin{itemize}
    \item Tests if the given key has been defined as a glossary entry key.
  \end{itemize}
\end{itemize}

3.3. Plurals

Some languages, such as English, have a general rule that plurals are formed from the singular with a suffix appended. This isn’t an absolute rule. There are plenty of exceptions (for example, geese, children, churches, elves, fairies, sheep). The \texttt{glossaries} package allows the \texttt{plural} key to be optional when defining entries. In some cases a plural may not make any sense (for example, the term is a symbol) and in some cases the plural may be identical to the singular.

To make life easier for languages where the majority of plurals can simply be formed by appending a suffix to the singular, the \texttt{glossaries} package lets the \texttt{plural} field default to the value of the \texttt{text} field with \texttt{\glspluralsuffix} appended. This command is defined to be just the letter “s”. This means that the majority of terms don’t need to have the \texttt{plural} supplied as well, and you only need to use it for the exceptions.

For languages that don’t have this general rule, the \texttt{plural} field will always need to be supplied, where needed.
3. Defining Entries

There are other plural fields, such as \texttt{firstplural}, \texttt{longplural} and \texttt{shortplural}. Again, if you are using a language that doesn’t have a simple suffix rule, you’ll have to supply the plural forms if you need them (and if a plural makes sense in the context).

If these fields are omitted, the glossaries package follows these rules:

- If \texttt{firstplural} is missing, then \texttt{\glspluralsuffix} is appended to the \texttt{first} field, if that field has been supplied. If the \texttt{first} field hasn’t been supplied but the \texttt{plural} field has been supplied, then the \texttt{firstplural} field defaults to the \texttt{plural} field. If the \texttt{plural} field hasn’t been supplied, then both the \texttt{plural} and \texttt{firstplural} fields default to the \texttt{text} field (or \texttt{name}, if no \texttt{text} field) with \texttt{\glspluralsuffix} appended.

- If the \texttt{longplural} field is missing, then \texttt{\glspluralsuffix} is appended to the \texttt{long} field, if the \texttt{long} field has been supplied.

- If the \texttt{shortplural} field is missing then, with the base glossaries acronym mechanism, \texttt{\acrpluralsuffix} is appended to the \texttt{short} field.

The last case is changed with glossaries-extra. With this extension package, the \texttt{shortplural} field defaults to the \texttt{short} field with \texttt{\abbrvpluralsuffix} appended unless overridden by category attributes. This suffix command is set by the abbreviation styles. This means that every time an abbreviation style is implemented, \texttt{\abbrvpluralsuffix} is redefined, see §4.1.2 for further details.

3.4. Entry Aliases

An entry can be made an alias of another entry using the \texttt{alias} key. The value should be the label of the other term. There’s no check for the other’s existence when the aliased entry is defined. This is to allow the possibility of defining the other entry after the aliased entry. (For example, when used with \texttt{bib2gls}.)

\begin{verbatim}
glsxtraliashook{⟨entry-label⟩}
\end{verbatim}

This hook is implemented when an entry is defined with the \texttt{alias} key set. It does nothing by default. The value of the \texttt{alias} field can be obtained with \texttt{\glsxtralias{⟨entry-label⟩}}.

If an entry \texttt{⟨entry-1⟩} is made an alias of \texttt{⟨entry-2⟩} then:

- If the \texttt{see} field wasn’t provided when \texttt{⟨entry-1⟩} was defined, the \texttt{alias} key will automatically trigger

\begin{verbatim}
\glssee{⟨entry-1⟩}{⟨entry-2⟩}
\end{verbatim}

- If the \texttt{hyperref} package has been loaded then \texttt{\gls{⟨entry-1⟩}} will link to \texttt{⟨entry-2⟩’s} target. (Unless the \texttt{targeturl} attribute has been set for \texttt{⟨entry-2⟩’s} category.)
• With record=off or record=hybrid, the noindex setting will automatically be triggered when referencing ⟨entry-1⟩ with commands like \gls or \glistext. This prevents ⟨entry-1⟩ from having a location list (aside from the cross-reference added with \glssee) unless it’s been explicitly indexed with \glsadd or if the indexing has been explicitly set using noindex=false. See §5.9.3 for adjusting the indexing hook.

Note that with record=only, the location list for aliased entries is controlled with bib2gls’s settings.

The value of the alias field can be accessed with \glsxtralias (see §5.9.2).

3.5. Setting or Updating Fields

See §5.11 for accessing field values and §5.15 for testing field values.

Modifications to fields only have an effect from that point onwards and may be localised to the current scope. If you are using docdef=true, any changes to the field values won’t be saved in the glsdefs file.

Some of these commands are subtly different from each other. For example, \glsfielddef (provided by the base glossaries package), \glsxtrdeffield and \GlsXtrSetField all assign a value to a field, but \glsfielddef requires that both the entry and the field exists (so it can’t be used to set an unknown internal field), \GlsXtrSetField requires that the entry exists (so it can be used to set an internal field that doesn’t have an associated key provided that the entry has been defined), and \glsxtrdeffield doesn’t perform any existence checks (which means that it can be used to assign internal fields before the entry is actually defined).

The commands described in this section don’t require the field to have an associated glossary entry key, so you need to be careful not to misspell the field labels.

Assigning or changing fields using the commands described here won’t alter related fields. For example, if you use the text key but not the plural key when you define an entry with \newglossaryentry, the plural key will automatically be set as well, but if you change the value of the text field after the entry has been defined, the plural field won’t be changed. Particular care is required if the field contributes in some way to the indexing information, as this information is typically initialised when the entry is first defined. This includes the sort and parent keys, which should not be changed after the entry has been defined.

With bib2gls, entries aren’t defined on the first \LaTeX run. This means that commands that test for existence will produce a warning and (within the document environment) the ?? unknown marker. For example:
3. Defining Entries

\documentclass{article}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[src=myentries,selection=all]
\begin{document}
Defining info
\glsxtrdeffield{sample}{info}{some information}.
Defining note
\GlsXtrSetField{sample}{note}{some note}.
Info: \glsxtrusefield{sample}{info}.
Note: \glsxtrusefield{sample}{note}.
\end{document}

On the first \LaTeX run, this produces:

\begin{quote}
Defining info . Defining note ??.
Info: some information. Note: .
\end{quote}

At this point the sample entry hasn’t been defined, so referencing it in \GlsXtrSetField results in a warning and the double question mark ?? unknown marker in the text. The field (note) isn’t saved, so nothing is shown when the field is referenced with \glsxtrusefield. Whereas \glsxtrdeffield does save the field with the label info associated with the label sample, even though the sample entry hasn’t actually been defined. The field can then be later obtained with \glsxtrusefield. Once \texttt{\bib2gls} has been run, the sample entry should now have its definition in the \texttt{glstex} file, which is loaded by \GlsXtrLoadResources and the note field can be set.

\begin{quote}
\glsxtrdeffield{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}
\end{quote}

This uses \texttt{etoolbox}'s \texttt{\csdef} command to locally set the field given by its internal label \texttt{(field-label)} to \texttt{(value)} for the entry identified by \texttt{(entry-label)}. No existence check is performed.

\begin{quote}
\glsxtrdeffield{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}
\end{quote}

This is like \glsxtrdeffield but (protected) fully expands the value before assigning it to the field.

\begin{quote}
\glsxtrapptocsvfield{⟨entry-label⟩}{⟨field-label⟩}{⟨element⟩}
\end{quote}

38
3. Defining Entries

This command is designed for fields that should contain comma-separated lists. If the field hasn’t been defined, this behaves like \texttt{\glsxtrdeffield} otherwise it will append a comma followed by \texttt{⟨element⟩} (unexpanded) to the field value. No existence check is performed. This field can be iterated over using \texttt{\glsxtrforcsvfield} or formatted using \texttt{\glsxtrfieldformatcsvlist}. See §5.13 for further details.

\begin{verbatim}
\glsxtrfieldlistadd{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}
\end{verbatim}

Appends the given value to the given entry’s field (identified using the field’s internal label) using etoolbox’s \texttt{\listcsadd}. The field value can later be iterated over using \texttt{\glsxtrfielddolistloop} or \texttt{\glsxtrfieldforlistloop}.

\begin{verbatim}
\glsxtrfieldlistgadd{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}
\end{verbatim}

As above but uses \texttt{\listcsgadd} to make a global change.

\begin{verbatim}
\glsxtrfieldlisteadd{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}
\end{verbatim}

As above but uses \texttt{\listcseadd} to expand the value.

\begin{verbatim}
\glsxtrfieldlistxadd{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}
\end{verbatim}

As above but uses \texttt{\listcsxadd} to make a global change.

\begin{verbatim}
\glsxtrsetfieldifexists{⟨entry-label⟩}{⟨field-label⟩}{⟨code⟩}
\end{verbatim}

This is used by the commands \texttt{\GlsXtrSetField}, \texttt{\gGlsXtrSetField}, \texttt{\xGlsXtrSetField}, \texttt{\eGlsXtrSetField}, \texttt{\GlsXtrLetField}, \texttt{\csGlsXtrLetField} and \texttt{\GlsXtrLetFieldToField} to produce an error (or warning with \texttt{undefaction=warn}) if the entry doesn’t exist. This can be redefined to add extra checks (for example, to prohibit changing certain fields).

\begin{verbatim}
\GlsXtrSetField{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}
\end{verbatim}

This uses etoolbox’s \texttt{\csdef} command to locally set the field given by its internal label \texttt{⟨field-label⟩} to \texttt{⟨value⟩} for the entry identified by \texttt{⟨entry-label⟩}.

This command is written to the \texttt{glstex} file by \texttt{bib2gls} to set fields that don’t have a
3. Defining Entries

corresponding key.

\GlsXtrSetField{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}

This is like \GlsXtrSetField but uses a global assignment.

\eGlsXtrSetField{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}

This is like \GlsXtrSetField but (protected) fully expands the value first.

\xGlsXtrSetField{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}

This is like \eGlsXtrSetField but uses a global assignment.

\GlsXtrLetField{⟨entry-label⟩}{⟨field-label⟩}{⟨cs⟩}

This uses etoolbox’s \cslet command to locally set the field given by its internal label ⟨field-label⟩ to ⟨cs⟩ for the entry identified by ⟨entry-label⟩.

\csGlsXtrLetField{⟨entry-label⟩}{⟨field-label⟩}{⟨cs-name⟩}

This uses etoolbox’s \csletcs command to locally set the field given by its internal label ⟨field-label⟩ to the control sequence given by ⟨cs-name⟩ for the entry identified by ⟨entry-label⟩.

\GlsXtrLetFieldToField{⟨entry1-label⟩}{⟨field1-label⟩}{⟨entry2-label⟩}{⟨field2-label⟩}

This assigns the field identified by its internal label ⟨field1-label⟩ for the entry identified by ⟨entry1-label⟩ to the value of the field identified by ⟨field2-label⟩ for the entry identified by ⟨entry2-label⟩.
4. Abbreviations

The acronym mechanism implemented by the base glossaries package is insufficiently flexible for some documents. The glossaries-extra package provides a completely different mechanism to deal with abbreviations in a more flexible manner. The two methods are incompatible. However, the glossaries-extra package provides predefined styles that emulate the appearance of the styles provided by the base package. If you have previously used just the base glossaries package, consult Table 4.2 on page 62 for the closest matching abbreviation style.

4.1. Defining Abbreviations

Abbreviations are defined using:

\newabbreviation\{⟨options⟩\}{⟨entry-label⟩}{⟨short⟩}{⟨long⟩}

where ⟨entry-label⟩ is the entry’s label (used in commands like \gls), ⟨short⟩ is the short form (the abbreviation) and ⟨long⟩ is the long form (what the abbreviation is short for). The optional argument ⟨options⟩ may be used to set additional keys (as per the options list in \newglossaryentry), such as type or category.

This command internally uses \newglossaryentry and sets the type to \glsxtrabbrv-type and the category to abbreviation. The category (see §10) determines the abbreviation style. The style for a particular category is set using \setabbreviationstyle. If the optional argument is omitted, the abbreviation category is assumed (see §4.5 for further details).

The following example document sets up three different abbreviation styles: long-short-sc for the abbreviation category, long-only-short-only for the custom genus category, and short-nolong for the custom common category. Note that the custom title category doesn’t have an associated style.

\setabbreviationstyle{long-short-sc}
\setabbreviationstyle[genus]{long-only-short-only}
\setabbreviationstyle[common]{short-nolong}
\newabbreviation{xml}{xml}{extensible markup language}
\newabbreviation{xml}{xml}{extensible markup language}
\newabbreviation[category={genus}]{clostridium}{C.}{Clostridium}
\newabbreviation[category={genus}]{myristica}{M.}{Myristica}
4. Abbreviations

```
\begin{document}
First use: \gls{xml}, \gls{clostridium}, \gls{myristica}, \gls{html}, \gls{dr}.

Next use: \gls{xml}, \gls{clostridium}, \gls{myristica}, \gls{html}, \gls{dr}.
\end{document}
```

Example 1: Multiple abbreviation styles

First use: extensible markup language (XML), Clostridium, Myristica, HTML, Doctor (Dr).
Next use: XML, C., M., HTML, Dr.

If the category doesn’t have an associated style, the style for the abbreviation category will be used, as with the dr entry above, which uses the long-short-sc style because no style has been associated with its custom title category.

There are two categories that have an abbreviation style set by default: abbreviation and acronym. These are initialised as follows:

```
\setabbreviationstyle{long-short}
\setabbreviationstyle[acronym]{short}
```

This means that abbreviations defined with the default abbreviation category will show the long form followed by the short form in parentheses on first use, and those defined with the category set to acronym will only show the short form (that is, the long form won’t be shown on first use).

To make it easier to migrate a file containing entries defined with \newacronym, the glossaries-extra package redefines \newacronym to do:

```
\newabbreviation[type=\acronymtype,category=acronym,(options)]{(entry-label)}{(short)}{(long)}
```

Note that this sets the category to acronym, which means that any abbreviations defined with \newacronym will use the short style by default. If you want to use a different style, you need to set the abbreviation style for the acronym category. For example, to use the long-short style:
This must be placed before the first instance of \newacronym.

You can’t use \setacronymstyle with glossaries-extra.

If you have defined any acronym styles with \newacronymstyle, you will have to migrate them over to \newabbreviationstyle. However, most of the predefined abbreviation styles are flexible enough to adapt to common abbreviation formats. It is possible to revert \newacronym back to using the base glossaries package’s acronym mechanism (§4.6), but it should generally not be necessary.

Terms defined with \newabbreviation (and \newacronym) can be referenced in the main document text using commands like \gls. (If you want to use shortcut commands like \ac, use the shortcuts=ac package option.) Remember that you can use the prereset and preunset options to reset or unset the first use flag (see §5.10). Alternatively, you can use the commands described in §4.3. For headings and captions, see §5.3.2.

Avoid using \glsfirst, \glsfirstplural, \glstext and \glsplural with abbreviations. Many of the abbreviation styles are too complex to work with these commands (particularly the case-changing variants or with the ⟨insert⟩ final optional argument or with innertextformat). Instead, use commands like \gls, \glsxtrshort, \glsxtrlong and \glsxtrfull.

4.1.1. Abbreviation Fields: long and short

The ⟨short⟩ and ⟨long⟩ arguments are internally assigned with the short and long keys (so don’t use those keys in ⟨options⟩), but the short and long values may first be modified by category attributes, such as markwords or markshortwords. As with other entries, avoid nested links (see §5.4). This means avoid using the \gls-like and \glstext-like commands within ⟨short⟩ and ⟨long⟩.

If an abbreviation can be formed by combining other entries, consider using the multi (compound) entry function (see §7).

4.1.2. Abbreviation Fields: longplural and shortplural

The longplural key defaults to ⟨long⟩\glspluralsuffix and the shortplural key defaults to ⟨short⟩\abbrvpluralsuffix. The aposplural attribute will instead set the short-
4. Abbreviations

plural to (short) ‘\abbrvpluralsuffix and the noshortplural attribute will set short-plural to just (short) (see §10). If these values are not appropriate, you will need to explicitly set the longplural and shortplural keys in ⟨options⟩.

The short plural suffix \abbrvpluralsuffix is redefined by the abbreviation style. Some styles, such as the long-short style, simply redefine \abbrvpluralsuffix to just:

\glsxtrabbrvpluralsuffix \textit{initial: \glspluralsuffix}

which is defined to \glspluralsuffix.

Some styles, such as the long-short-sc style, redefine \abbrvpluralsuffix to include code to counteract the formatting of the abbreviation.

If you want to change the default short plural suffix, you need to redefine \glsxtrabbrvpluralsuffix not \abbrvpluralsuffix. If you don’t want the suffix added, then set the noshortplural attribute to true.

4.1.3. Abbreviation Fields: name and description

The name key is set according to the abbreviation style. There should not be any need to explicitly set it. Some styles require the description key to be set in ⟨options⟩, but other styles will set the description to the long form.

4.1.4. Abbreviation Fields: type

Abbreviations can be assigned to a particular glossary using the type key in ⟨options⟩. The default for \newabbreviation is:

\glsxtrabbrvtype \textit{initial: \glsdefaulttype}

This is initialised to \glsdefaulttype (the default glossary), but the abbreviations package option will redefine it to abbreviations.

The default type for \newacronym is:

\acronymtype \textit{initial: \glsdefaulttype}

This is initialised to \glsdefaulttype, but the acronyms package option will redefine it to acronym.
4. Abbreviations

4.1.5. General Hooks

The following are general purpose hooks used within \newabbreviation. Note that there are additional hooks that are used by the abbreviation styles (see §4.5.3.1).

\glsxtrnewabbrevpresethook{(options)}{(label)}{(short)}

This hook is provided for further customisation, if required. It's implemented before the entry is defined (before the shortplural and longplural keys supplied in (options) are parsed). Does nothing by default. The arguments are a legacy throwback to old versions that didn't have \glsxtrorgshort.

\newabbreviationhook

This hook is performed just before the entry is defined. Does nothing by default.

4.2. Examples: makeindex vs bib2gls

Example document using makeindex:

\documentclass{article}
\usepackage{glossaries-extra}
\makeglossaries
\newglossaryentry{sample}{name={sample},description={an example}}
\newabbreviation{xml}{XML}{extensible markup language}
\newacronym{nasa}{NASA}{National Aeronautics and Space Administration}
\begin{document}
First use: \gls{sample}, \gls{xml} and \gls{nasa}.
Next use: \gls{sample}, \gls{xml} and \gls{nasa}.
\printglossaries
\end{document}
Example 2: `\newabbreviation` vs `\newacronym` vs `\newglossaryentry`

First use: sample, extensible markup language (XML) and NASA. Next use: sample, XML and NASA.

### Glossary

**NASA** National Aeronautics and Space Administration 1

**sample** an example 1

**XML** extensible markup language 1

Note that the long form of NASA isn’t displayed on the first use of `\gls{nasa}`. This is because the acronym category uses the short style by default.

In the above example, all entries are placed in the main (default) glossary. The package options `abbreviations` and `acronyms` can be used to split them off into separate glossaries.

If you use `bib2gls`, the analogous bib entry types are `@abbreviation` and `@acronym`. The above example can be rewritten to use `bib2gls`:

```latex
\documentclass{article}
\begin{filecontents*}{\jobname.bib}
@entry{sample, 
  name={sample}, 
  description={an example}}
}\end{filecontents*}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources
\begin{document}
First use: \gls{sample}, \gls{xml} and \gls{nasa}. Next use: \gls{sample}, \gls{xml} and \gls{nasa}.
\end{document}
```
4. Abbreviations

Example 3: @abbreviation vs @acronym vs @entry

First use: sample, extensible markup language (XML) and NASA. Next use: sample, XML and NASA.

Glossary

NASA National Aeronautics and Space Administration 1
sample an example 1
XML extensible markup language 1

4.3. Referencing (Using) Abbreviations

Since \newabbreviation internally uses \newglossaryentry, you can reference abbreviations with the \gls-like commands as with other entries. Remember that you can use the \prerset and \preunset options to reset or unset the first use flag (see §5.10).

In general it’s best not to use \glsfirst, \glsfirstplural, \glstext, \glplural or their case-changing variants as many of the abbreviation styles are too complicated for those commands. If you specifically want the full form, use \gls with \prerset or use \glsxtr-full. If you specifically want the short form for a particular instance, use \gls with \preunset or use \glsxtrshort. If you only want the long form for a particular instance, use \glsxtrlong.

If you never want the short form with \gls, use one of the “noshort” styles, such as long-noshort. If you never want the long form with \gls, use one of the “nolong” styles, such as short-nolong.

If you need to use abbreviations in headings or captions, use commands like \glsfmtshort and \glsfmtlong (see §5.3.2). Commands like \glsentryname are likely to contain non-expandable content.

Example:
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\makeglossaries
\newabbreviation{svm}{SVM}{support vector machine}
\newabbreviation{html}{HTML}{hypertext markup language}

\begin{document}
\tableofcontents
\section{Introducing the \glsfmtlong{svm}}
First use: \gls{svm}.
Next use: \gls{svm}.

\section{Introducing \gls{html} (incorrect)}
First use (not!): \gls{html}.
Next use: \gls{html}.
\glsreset{html}

\section{Introducing \glsxtrshort{html} (incorrect)}
First use: \gls{html}.
Next use: \gls{html}.
\glsreset{html}

\section{Introducing \glsfmtshort{html}}
First use: \gls{html}.
Next use: \gls{html}.
\printglossaries
\end{document}
Example 4: Referencing an abbreviation (with hyperref)

Contents

1 Introducing the support vector machine 1
2 Introducing hypertext markup language (HTML) (incorrect) 1
3 Introducing HTML (incorrect) 1
4 Introducing HTML 1

Glossary

HTML hypertext markup language 1
SVM support vector machine 1

In the above example, compare the first section heading (which references an abbreviation with \glsfmtlong) with the second section heading (which references an abbreviation with \gls). Note that the first use of the html entry actually occurs in the table of contents, which results in the full form showing in the table of contents, but only the abbreviation is shown.
in the actual section 2 heading. The PDF bookmark shows the entry label (html) not the abbreviation (HTML). There is also a nested link for section 2 in the table of contents. In some PDF viewers (such as Okular), this will lead to section 2 but, in others (such as Evince), it will lead to the HTML entry target in the glossary. Similarly for section 3.

As with the base glossaries package, the unformatted short and long forms can be obtained with \glsentryshort and \glsentrylong or, for the plural forms, \glsentryshortpl and \glsentrylongpl. These are analogous to commands like \glsentrytext and may be used in expandable contexts. The sentence case versions (\Glsentryshort, \Glsentrylong, \Glsentryshort, and \Glsentrylong) are all robust in glossaries v4.49 and lower. As from glossaries v4.50, they can expand in PDF bookmarks, but outside of PDF bookmarks they will expand to a robust internal command.

Don’t use the base glossaries package’s acronym commands, such as \acrshort. These aren’t compatible with \newabbreviation.

Each abbreviation style has a display full form, which is the format produced with the first use of \gls, and an inline full form, which is the format produced by \glsxtrfull. For some styles, such as long-short, the display and inline forms are identical.

The example below demonstrates the difference between the first use of \gls compared with the inline full form for the footnote abbreviation style. The example also uses \gls-first to demonstrate that it produces an undesirable result with the selected abbreviation style.
In the above, the first use of \gls puts the long form in the footnote but correctly inserts the final optional argument before the footnote marker. The inline full form (obtained with \glsxtrfull) doesn’t use a footnote, but instead shows the long form in parentheses after the short form. The insert material is correctly placed after the short form. Compare this with the final line, which uses \glsfirst. This shows the long form in the footnote, but the inserted material can’t be shifted before the footnote marker, which results in the strange “NASA’s”.

The following commands are included in the set of \glistext-like commands. They have the same options as \glistext and don’t change the first use flag. They will index (unless noindex is used), create a hyperlink (if enabled), and use the post-link hook.

\glsxlabel\[⟨options⟩]\{⟨entry-label⟩}\{⟨insert⟩\] modifiers: * + ⟨alt-mod⟩
Displays the short form using the abbreviation style’s formatting.
4. Abbreviations

As above, but sentence case version.

\GLSxtrshort[{\textit{options}}]{\langle entry-label\rangle}{\langle insert\rangle} \quad \text{modifiers: } * + \langle \textit{alt-mod}\rangle

As above, but all caps version.

\GLSxtrshortpl[{\textit{options}}]{\langle entry-label\rangle}{\langle insert\rangle} \quad \text{modifiers: } * + \langle \textit{alt-mod}\rangle

Displays the short plural form using the abbreviation style’s formatting.

\GLSxtrshortpl[{\textit{options}}]{\langle entry-label\rangle}{\langle insert\rangle} \quad \text{modifiers: } * + \langle \textit{alt-mod}\rangle

As above, but sentence case version.

\GLSxtrshortpl[{\textit{options}}]{\langle entry-label\rangle}{\langle insert\rangle} \quad \text{modifiers: } * + \langle \textit{alt-mod}\rangle

As above, but all caps version.

\GLSxtrpl[{\textit{options}}]{\langle entry-label\rangle}{\langle insert\rangle} \quad \text{modifiers: } * + \langle \textit{alt-mod}\rangle

Displays the long form using the abbreviation style’s formatting. As from v1.49, this command simulates first use by defining \glsxtrifwasfirstuse to do its first argument. This is done via the command:

\glsxtrsetlongfirstuse{\langle entry-label\rangle}

which is defined as:

\begin{verbatim}
\newcommand{\glsxtrsetlongfirstuse}[1]{% 
  \let\glsxtrifwasfirstuse@firstoftwo%
}\
\end{verbatim}

This command takes the entry label as the argument, which is ignored by default.

To restore the original behaviour, redefine this command as follows:
4. Abbreviations

\renewcommand{\glsxtrsetlongfirstuse}{1}{\%
\letcs\glsxtrifwasfirstuse{\@secondoftwo}{\%

This command is also used by the case-changing and plural variants listed below.

\Glsxtrlong[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As above, but sentence case version.

\GLSxtrlong[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As above, but all caps version.

\glsxtrlongpl[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

Displays the long plural form using the abbreviation style’s formatting.

\Glsxtrlongpl[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As above, but sentence case version.

\GLSxtrlongpl[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As above, but all caps version.

\glsxtrfull[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

Displays the inline full form using the abbreviation style’s formatting. Depending on the style, this may not be the same as the text produced with the first use of \gls.
4. Abbreviations

As above, but sentence case version.

\GLSxtrfull{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩}

As above, but all caps version.

\glsxtrfullpl{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩}

Displays the inline full plural form using the abbreviation style’s formatting. Depending on the style, this may not be the same as the text produced with the first use of \glsxpl.

\Glsxtrfullpl{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩}

As above, but sentence case version.

\GLSxtrfullpl{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩}

As above, but all caps version.

\glsxtrsetupfulldefs

This hook is used within \glsxtrfull, \glsxtrfullpl and the case-changing variations to initialise \glsxtrifwasfirstuse in case it’s required in the post-link hook. The default definition is to simulate first use. Note that changing this can cause unexpected results with abbreviation styles that set the post-link hook, such as short-postlong-user.

\glsxtrfullsaveinsert{⟨entry-label⟩}{⟨insert⟩}

This hook is used within \glsxtrfull, \glsxtrfullpl and the case-changing variations to initialise the \glsinsert placeholder. The default definition is to use \glsxtrsaveinsert. If the insert isn’t saved, it can’t be used within the post-link hook for the \glsxtrfull etc. This affects the behaviour of the “post-hyphen” abbreviation styles, such as long-hyphen-postshort-hyphen.

4.3.1. Prefixes

If you are using the glossaries-prefix package (which can be loaded via the prefix package option), then there are commands similar to \glsxtrshort and \glsxtrlong that insert
the corresponding prefix and separator at the front if the short or long form, if the prefix has been set and is non-empty. In all cases, the separator is \gls{prefixsep}, as with \gls{pl}.

These commands require glossaries-pre.

\glsxtrshort{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As \glsxtrshort but inserts the prefix field and separator, if the prefix value is set and non-empty.

\glsxtrshortpl{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As \glsxtrshortpl but inserts the prefixplural field and separator, if the prefixplural value is set and non-empty.

\glsxtrlong{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As \glsxtrlong but all caps.
4. Abbreviations

As \texttt{\glssxtrlong} but inserts the \texttt{prefixfirst} field and separator, if the \texttt{prefixfirst} value is set and non-empty.

\begin{verbatim}
\texttt{\Pglssxtrlong\langle options\rangle\{entry-label\}\{insert\}} modifiers: * + (alt-mod)
\end{verbatim}

As \texttt{\pglsxtrlong} but sentence case.

\begin{verbatim}
\texttt{\PGLSxtrlong\langle options\rangle\{entry-label\}\{insert\}} modifiers: * + (alt-mod)
\end{verbatim}

As \texttt{\pglsxtrlong} but all caps.

\begin{verbatim}
\texttt{\pglsxtrlongpl\langle options\rangle\{entry-label\}\{insert\}} modifiers: * + (alt-mod)
\end{verbatim}

As \texttt{\glssxtrlongpl} but inserts the \texttt{prefixfirstplural} field and separator, if the \texttt{prefixfirstplural} value is set and non-empty.

\begin{verbatim}
\texttt{\Pglssxtrlongpl\langle options\rangle\{entry-label\}\{insert\}} modifiers: * + (alt-mod)
\end{verbatim}

As \texttt{\pglsxtrlongpl} but sentence case.

\begin{verbatim}
\texttt{\PGLSxtrlongpl\langle options\rangle\{entry-label\}\{insert\}} modifiers: * + (alt-mod)
\end{verbatim}

As \texttt{\pglsxtrlongpl} but all caps.

4.3.2. Abbreviation Shortcut Commands

The abbreviation shortcut commands can be enabled using the \texttt{shortcuts=abbreviations} package option (or \texttt{shortcuts=abbr} or \texttt{shortcuts=ac}). The provided shortcut commands listed in Table 4.1 on the next page. Note that \texttt{\glssxtrenablerecordcount} will switch the shortcuts that use the \texttt{\cgls}-like commands to the corresponding \texttt{\rgls}-like command.

4.4. Tagging Initials

Initial tagging allows you to highlight the initials that form the abbreviation when the long form is shown in the glossary.

\begin{verbatim}
\texttt{\GlsXtrEnableInitialTagging\langle categories\rangle\{cs\}} modifier: *
\end{verbatim}
4. Abbreviations

Table 4.1.: Abbreviation Shortcut Commands

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>Shortcut</th>
<th>Equivalent Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ab</td>
<td>\ac</td>
<td>\cglst</td>
</tr>
<tr>
<td>\abp</td>
<td>\acp</td>
<td>\cglsp1</td>
</tr>
<tr>
<td>\as</td>
<td>\acs</td>
<td>\glsxtrshort</td>
</tr>
<tr>
<td>\asp</td>
<td>\acsp</td>
<td>\glsxtrshortpl</td>
</tr>
<tr>
<td>\al</td>
<td>\acl</td>
<td>\glsxtrlong</td>
</tr>
<tr>
<td>\alp</td>
<td>\aclp</td>
<td>\glsxtrlongpl</td>
</tr>
<tr>
<td>\af</td>
<td>\acf</td>
<td>\glsxtrfull</td>
</tr>
<tr>
<td>\afp</td>
<td>\acfpl</td>
<td>\glsxtrfullpl</td>
</tr>
<tr>
<td>\Ab</td>
<td>\Ac</td>
<td>\cglst</td>
</tr>
<tr>
<td>\Abp</td>
<td>\Acpl</td>
<td>\cglsp1</td>
</tr>
<tr>
<td>\As</td>
<td>\Acsp</td>
<td>\glsxtrshortpl</td>
</tr>
<tr>
<td>\Asp</td>
<td>\Acsppl</td>
<td>\glsxtrlongpl</td>
</tr>
<tr>
<td>\Al</td>
<td>\Acpl</td>
<td>\glsxtrlongpl</td>
</tr>
<tr>
<td>\Alp</td>
<td>\Acplpl</td>
<td>\glsxtrfullpl</td>
</tr>
<tr>
<td>\Af</td>
<td>\Acfl</td>
<td>\glsxtrfull</td>
</tr>
<tr>
<td>\Afp</td>
<td>\Acflpl</td>
<td>\glsxtrfullpl</td>
</tr>
<tr>
<td>\AB</td>
<td>\AC</td>
<td>\cGLS</td>
</tr>
<tr>
<td>\ABP</td>
<td>\ACP</td>
<td>\cGLSp1</td>
</tr>
<tr>
<td>\AS</td>
<td>\ACSP</td>
<td>\glsxtrshort</td>
</tr>
<tr>
<td>\ASP</td>
<td>\ACSPpl</td>
<td>\glsxtrshortpl</td>
</tr>
<tr>
<td>\AL</td>
<td>\ACL</td>
<td>\glsxtrlong</td>
</tr>
<tr>
<td>\ALP</td>
<td>\ACLP</td>
<td>\glsxtrlongpl</td>
</tr>
<tr>
<td>\AF</td>
<td>\ACF</td>
<td>\glsxtrfull</td>
</tr>
<tr>
<td>\AFP</td>
<td>\ACFP</td>
<td>\glsxtrfullpl</td>
</tr>
<tr>
<td>\newabbr</td>
<td>\newabbr</td>
<td>\newabbreviation</td>
</tr>
</tbody>
</table>
4. Abbreviations

\GlsXtrEnableInitialTagging must be placed before the abbreviations are defined.

This command (robustly) defines \(<cs>\) (a control sequence) to accept a single argument, which is the letter (or letters) that needs to be tagged. The normal behaviour of \(<cs>\) within the document is to simply do its argument, but in the glossary it’s activated for those categories that have the \textit{tagging} attribute set to “true”. For those cases it will use:

\glsxtrtagfont{⟨text⟩}

This command defaults to \underline{⟨text⟩} but may be redefined as required.

The control sequence \(<cs>\) can’t already be defined when used with the unstarred version of \GlsXtrEnableInitialTagging for safety reasons. The starred version will overwrite any previous definition of \(<cs>\). As with redefining any commands, ensure that you don’t redefine something important.

The first argument of \GlsXtrEnableInitialTagging is a comma-separated list of category names. The \textit{tagging} attribute will automatically be set to \textit{true} for those categories. You can later set this attribute for other categories (see §10) but this must be done before the glossary is displayed.

For example, the following uses initial tagging for both the \textit{acronym} and \textit{abbreviation} categories. The custom command \texttt{\itag} is defined as the tagging command.

\begin{verbatim}
\makeglossaries
\GlsXtrEnableInitialTagging{acronym,abbreviation}{\itag}
\setabbreviationstyle[acronym]{short-nolong-desc}
\newacronym{description={a system for detecting the location and speed of ships, aircraft, etc, through the use of radio waves}}{radar}{radar}{\itag{ra}dio \itag{d}etection \itag{a}nd \itag{r}anging}
\newabbreviation{xml}{XML}{e\itag{x}tensible \ntag{m}arkup \ntag{l}anguage}
\newabbreviation[category={other}]{tne}{\ntag{t}agging \ntag{n}ot \ntag{e}nabled}
\end{verbatim}
4. Abbreviations

Example 6: Tagging abbreviation initials

First use: radar, extensible markup language (XML), tagging not enabled (TNE).
Long form only: radio detection and ranging, extensible markup language, tagging not enabled.

Glossary

radar (radio detection and ranging) a system for detecting the location and speed of ships, aircraft, etc, through the use of radio waves 1

TNE tagging not enabled 1

XML extensible markup language 1

The underlining of the tagged letters only occurs in the glossary and then only for entries with the tagging attribute set.

4.5. Abbreviation Styles

The style for a particular category is set using:

\setabbreviationstyle[(category)]{(style-name)}

If the (category) is omitted, abbreviation is assumed. Remember that \newacronym sets the category to acronym so with \newacronym you need to change the style with:

\setabbreviationstyle[acronym]{(style-name)}
4. Abbreviations

The abbreviation style must be set before the abbreviation with the corresponding category is defined. If you are using bib2gls, the style must be set before \GlsXtr-LoadResources.

The style associated with the abbreviation category will be used if an abbreviation is defined with a category that doesn’t have an associated style.

Once you have defined an abbreviation with a given category, you can’t subsequently change the style for that category. You can’t have more than one style per category. The default style for the abbreviation category is long-short and the default style for the acronym category is short-nolong.

In the example below, the custom latin category doesn’t have an associated abbreviation style, so it uses the style assigned to the abbreviation category, not the acronym category. The only reason that the “radar” abbreviation (defined with \newacronym) uses the style associated with the acronym category is because the default definition of \newacronym sets category={acronym}.

\\usepackage[T1]{fontenc}  
\usepackage{glossaries-extra}  
\setabbreviationstyle{long-short-sc}  
\newabbreviation{html}{html}  
{hypertext markup language}  
\setabbreviationstyle[acronym]  
{footnote}  
\newacronym{radar}{radar}  
{radio detection and ranging}  
\newacronym[category={latin}]{ibid}{ibid}{ibidem}  
\begin{document}  
\gls{html}, \gls{radar} and \gls{ibid}.  
\printunsrtglossaries  
\end{document}

Example 7: Category without an associated abbreviation style

hypertext markup language (HTML), radar\textsuperscript{1} and ibidem (IBID).

Glossary

\textbf{HTML} hypertext markup language

\textbf{radar} radio detection and ranging

\textbf{IBID} ibidem

4.5.1. Predefined Abbreviation Styles

There are two types of abbreviation styles: those that treat the abbreviation as a regular entry (so that \gls uses \glsgenentryfmt and is encapsulated with \glsxtrregularfont) and those that don’t treat the abbreviation as a regular entry (so that \gls uses \glsxtr-
genabbrvfmt and is encapsulated with \glsxtrabbreviationfont). See §5.5.5 for further details of those commands.

The non-regular abbreviation styles allow for more complex formats than the regular styles.

The regular entry abbreviation styles set the regular attribute to true for the category assigned to each abbreviation with that style. This means that on first use, \gls uses the value of the first field and on subsequent use \gls uses the value of the text field (and analogously for the plural and case-changing versions).

The non-regular abbreviation styles don’t set the regular attribute, unless it has already been set, in which case it will be changed to false. The first and text fields (and their plural forms) are set, but they aren’t used by commands like \gls, which instead use formatting commands, such as \glsxtrfullformat and \glsxtrsubsequentfmt, which are defined by the style.

In both cases, the first use of \gls may not match the text produced by \glsfirst (and likewise for the plural and case-changing versions).

The short and long fields are set as appropriate and may be accessed through commands like \glsxtrshort and \glsxtrlong. These may appear slightly differently to the way the short or long form is displayed within \gls, depending on the style.

The sample file sample-abbr-styles.pdf demonstrates all predefined styles described here.

For the “sc” styles that use \textsc, be careful about your choice of fonts as some only have limited support. For example, you may not be able to combine bold and small-caps. If you’re using pdfLaTeX, I recommend that you at least use the fontenc package with the T1 option or something similar.

The predefined styles have helper commands to make it easier to modify the format. These are described in §4.5.1.3.

Table 4.2 lists the nearest equivalent glossaries-extra abbreviation styles for the predefined acronym styles provided by glossaries, but note that the new styles use different formatting commands.

The example documents used to illustrate the predefined styles in the sub-sections below are all in the form (document class and options may vary):

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
4. Abbreviations

Table 4.2.: Base Acronym Styles \setacroystyle{⟨base-style-name⟩} Verses New Abbreviation Styles \setabbreviostyle{⟨category⟩}{⟨new-style-name⟩}

<table>
<thead>
<tr>
<th>Base Style Name</th>
<th>New Style Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>long-sc-short</td>
<td>long-short-sc</td>
</tr>
<tr>
<td>long-sm-short</td>
<td>long-short-sm</td>
</tr>
<tr>
<td>long-sp-short</td>
<td>long-short</td>
</tr>
<tr>
<td>short-long</td>
<td>short-long</td>
</tr>
<tr>
<td>sc-short-long</td>
<td>short-sc-long</td>
</tr>
<tr>
<td>sm-short-long</td>
<td>short-sm-long</td>
</tr>
<tr>
<td>long-short-desc</td>
<td>long-short-desc</td>
</tr>
<tr>
<td>long-sc-short-desc</td>
<td>long-short-sc-desc</td>
</tr>
<tr>
<td>long-sm-short-desc</td>
<td>long-short-sm-desc</td>
</tr>
<tr>
<td>long-sp-short-desc</td>
<td>long-short-desc</td>
</tr>
<tr>
<td>short-long-desc</td>
<td>short-long-desc</td>
</tr>
<tr>
<td>sc-short-long-desc</td>
<td>short-sc-long-desc</td>
</tr>
<tr>
<td>sm-short-long-desc</td>
<td>short-sm-long-desc</td>
</tr>
<tr>
<td>dua</td>
<td>long-noshort</td>
</tr>
<tr>
<td>dua-desc</td>
<td>long-noshort-desc</td>
</tr>
<tr>
<td>footnote</td>
<td>short-footnote</td>
</tr>
<tr>
<td>footnote-sc</td>
<td>short-sc-footnote</td>
</tr>
<tr>
<td>footnote-sm</td>
<td>short-sm-footnote</td>
</tr>
<tr>
<td>footnote-desc</td>
<td>short-footnote-desc</td>
</tr>
<tr>
<td>footnote-sc-desc</td>
<td>short-sc-footnote-desc</td>
</tr>
<tr>
<td>footnote-sm-desc</td>
<td>short-sm-footnote-desc</td>
</tr>
</tbody>
</table>
4. Abbreviations

\setabbreviationstyle{⟨style-name⟩}
\newabbreviation[⟨options⟩]{ex}{⟨short⟩}{long form}
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert].
\printunsrtglossaries
\end{document}

where ⟨style-name⟩ is the name of the abbreviation style, ⟨short⟩ is either “SHRT FM” or (for the small caps examples) “shrt fm”. The styles that require the description or user1 key to be set will include that in ⟨options⟩ otherwise the optional argument of \newabbreviation will be omitted. The examples with a style that requires \textsmaller will load relsize. The “hyphen” styles set the markwords and markshortwords attributes. Note that hyperref is loaded with the colorlinks option, so the hyperlink text will be red.

The naming scheme for abbreviation styles is as follows:

- ⟨field1⟩[-⟨modifier1⟩][-⟨field2⟩][post][⟨field2⟩][-⟨modifier2⟩]][-user]

This is for the parenthetical styles. The -⟨modifier⟩ parts may be omitted. These styles display ⟨field1⟩ followed by ⟨field2⟩ in parentheses. If ⟨field1⟩ or ⟨field2⟩ starts with “no” then that element is omitted from the display style (no parenthetical part) but is included in the inline style.

If post is present then ⟨field2⟩ is placed after the link text using the post-link hook. Note that this will use the singular form of ⟨field2⟩ by default, even if \glsp1 is used. The corresponding non-post style will use the matching form for ⟨field2⟩.

If the -⟨modifier⟩ part is present and is one of sc, sm or em, then the field has a font changing command applied to it.

The modifier -only indicates that field is only present according to whether or not the entry has been used.

The modifier -hyphen indicates the style will substitute inter-word spaces (that have been marked up with the markwords or markshortwords attributes) will be changed to spaces if the inserted material starts with a hyphen (but not for the set of \glsxtrshort and \glsxtrlong commands).

If the -user part is present, then the value of the field given by \glsxtruserinfo (user1), if set, is inserted into the parenthetical material.

Examples:
- long-noshort-sc: ⟨field1⟩ is the long form, the short form is set in small caps but omitted in the display style.
- long-em-short-em: both the long form and the short form are emphasized. The short form is in parentheses.
- long-short-em: the short form is emphasized but not the long form. The short form is in parentheses.
4. Abbreviations

- **long-short-user**: if the `user1` key has been set, this produces the style \(\langle long\rangle((short),(user1))\) otherwise it just produces \(\langle long\rangle((short))\).

- **long-hyphen-postshort-hyphen**: the short form and the inserted material (provided by the final optional argument of commands like \(\texttt{gls}\)) is moved to the post-link hook. The long form is formatted according to \(\texttt{glslonghyphenfont}\) (or \(\texttt{glsfirstlonghyphenfont}\) on first use). The short form is formatted according to \(\texttt{glsabbrvhyphenfont}\) (or \(\texttt{glsfirstabbrvhyphenfont}\) on first use).

• **\langle style\rangle-noreg**

Some styles set the *regular* attribute. In some cases, there's a version of the style that doesn't set this attribute. For example, *long-em-noshort-em* sets the `regular` attribute. The *long-em-noshort-em-noreg* style is a minor variation of that style that sets the attribute to `false`.

There are a few “noshort” styles, such as *long-hyphen-noshort-noreg*, where there isn't a corresponding regular version. This is because the style won’t work properly with the `regular` attribute set, but the naming scheme is maintained for consistency with the other “noshort” styles.

• **\langle field1\rangle-[\langle modifier1\rangle]-[post]footnote**

The display style uses \(\langle field1\rangle\) followed by a footnote with the other field in it. If `post` is present then the footnote is placed after the link text using the post-link hook. The inline style does \(\langle field1\rangle\) followed by the other field in parentheses.

If \(\langle modifier1\rangle\) is present, \(\langle field1\rangle\) has a font-changing command applied to it.

Examples:

- **short-footnote**: short form in the text with the long form in the footnote.
- **short-sc-postfootnote**: short form in smallcaps with the long form in the footnote outside of the link text.

Take care with the footnote styles. Remember that there are some situations where \(\texttt{\footnote}\) doesn't work.

• **\langle style\rangle-desc**

Like \(\langle style\rangle\) but the *description* key must be provided when defining abbreviations with this style.

Examples:

- **short-long-desc**: like short-long but requires a description.
- **short-em-footnote-desc**: like short-em-footnote but requires a description.
4. Abbreviations

Not all combinations that fit the above syntax are provided. Pre-version 1.04 styles that didn’t fit this naming scheme are either provided with a synonym (where the former name wasn’t ambiguous) or provided with a deprecated synonym (where the former name was confusing).

4.5.1.1. Regular Styles

The following abbreviation styles set the regular attribute to true for all categories that have abbreviations defined with any of these styles. This means that they are treated like ordinary entries and are encapsulated with \glsxtrregularfont not \glsxtrabbreviationfont. The \gls-like commands are formatted according to \glsgenentryfmt.

4.5.1.1.1. Short Styles

These styles only show the short form on both first use and subsequent use. See §4.5.1.3.1 and §4.5.1.3.5 for style commands.

\setabbreviationstyle{short-nolong}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}

Example 8: The short-nolong abbreviation style


Glossary

SHRT FM long form

The long form is formatted with \glslongdefaultfont for the \glsxtrlong set of commands.

The short form is formatted with \glsfirstabbrvdefaultfont within the full form and
4. Abbreviations

with \glsabbrvdefaultfont for subsequent use and for the \glsxtrshort set of commands.
The name is set to the short form (\glsxtrshortnolongname) and the description is set to the unencapsulated long form.

<table>
<thead>
<tr>
<th>abbrev</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>short</td>
<td>alias: short-nolong</td>
</tr>
</tbody>
</table>

A synonym for short-nolong.

<table>
<thead>
<tr>
<th>abbrev</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>short-nolong-desc</td>
<td></td>
</tr>
</tbody>
</table>

As short-nolong but the description must be supplied in the optional argument of \newabbreviation. Font variations are available with short-sc-nolong-desc, short-sm-nolong-desc and short-em-nolong-desc.

\setabbreviationstyle{short-nolong-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}

Example 9: The short-nolong-desc abbreviation style


Glossary

SHRT FM (long form) sample description

The name is set to the short form followed by the long form in parentheses (\glsxtrshortdescname), and the sort is set to just the short form.

<table>
<thead>
<tr>
<th>abbrev</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>short-desc</td>
<td>alias: short-nolong-desc</td>
</tr>
</tbody>
</table>
4. Abbreviations

A synonym for short-nolong-desc.

nolong-short

The same as short-nolong except for the inline full form, which shows the same parenthetical style as long-short (\glsxtrlongshortformat). Font variations are available with nolong-short-sc, nolong-short-sm and nolong-short-em.

\setabbreviationstyle{nolong-short}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}

Example 10: The nolong-short abbreviation style


Glossary

SHRT FM long form

short-sc-nolong

This style is like short-nolong but it uses \glsxtrscsuffix, \glsabbrvscfont and \gls-firstabbrvscfont (see §4.5.1.3.9).

\setabbreviationstyle{short-sc-nolong}
\newabbreviation{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}
Example 11: The short-sc-nolong abbreviation style


**Glossary**

**SHRT FM** long form

---

**short-sc**

alias: short-sc-nolong

A synonym for short-sc-nolong.

**short-sc-nolong-desc**

This style is like short-nolong-desc but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).

\setabbreviationstyle{short-sc-nolong-desc}
\newabbreviation[description={sample description}]{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}-insert. Next: \gls{ex}-insert. Full: \glsxtrfull{ex}-insert. First plural: \glspl[prereset]{ex}-insert. \printunsrtglossaries \end{document}

Example 12: The short-sc-nolong-desc abbreviation style


**Glossary**

**SHRT FM** (long form) sample description
4. Abbreviations

**short-sc-desc**

*alias: short-sc-nolong-desc*

A synonym for *short-sc-nolong-desc*.

**nolong-short-sc**

This style is like *nolong-short* but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).

```
\setabbreviationstyle{nolong-short-sc}
\newabbreviation{ex}{shrt fm}{long form}
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural:
\glsp{ex}[-insert].
\end{document}
```

**Example 13**: The *nolong-short-sc* abbreviation style


**Glossary**

*SHRT FM*  long form

**short-sm-nolong**

This style is like *short-nolong* but it uses \glsxtrsmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```
\setabbreviationstyle{short-sm-nolong}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural:
\glsp{ex}[-insert].
\end{document}
```
Example 14: The short-sm-nolong abbreviation style


Glossary

SHRT FM long form

short-sm alias: short-sm-nolong

A synonym for short-sm-nolong.

short-sm-nolong-desc

This style is like short-nolong-desc but it uses \glsxtrmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

\setabbreviationstyle{short-sm-nolong-desc}
\newabbreviation[description={sample description}]\{ex\}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}{-insert}. First plural: \glspl[prereset]{ex}{-insert}. \printunsrtglossaries \end{document}

Example 15: The short-sm-nolong-desc abbreviation style


Glossary

SHRT FM (long form) sample description
4. Abbreviations

**short-sm-desc**

| alias: short-sm-nolong-desc |

A synonym for **short-sm-nolong-desc**.

**nolong-short-sm**

This style is like **nolong-short** but it uses \glsxtrmsuffix, \glsabbrvsmfont and \glstfirstabbrvsmfont (see §4.5.1.3.9).

```latex
\setabbreviationstyle{nolong-short-sm}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}-insert. Next: \gls{ex}-insert. Full: \glsxtrfull{ex}-insert. First plural: \glsp[prereset]{ex}-insert. \printunsrtglossaries \end{document}
```

Example 16: The **nolong-short-sm** abbreviation style


**Glossary**

**SHRT FM** long form

**short-em-nolong**

This style is like **short-nolong** but it uses \glsxtremsuffix, \glsabbrvemfont and \glstfirstabbrvemfont (see §4.5.1.3.9).

```latex
\setabbreviationstyle{short-em-nolong}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}-insert. Next: \gls{ex}-insert. Full: \glsxtrfull{ex}-insert. First plural: \glsp[prereset]{ex}-insert. \printunsrtglossaries \end{document}
```
4. Abbreviations

\end{document}

Example 17: The short-em-nolong abbreviation style


\begin{glossary}
\textbf{SHRT FM} long form
\end{glossary}

\begin{itemize}
\item short-em \textit{alias: short-em-nolong}
\end{itemize}

A synonym for short-em-nolong.

\begin{itemize}
\item short-em-nolong-desc
\end{itemize}

This style is like short-nolong-desc but it uses \textbackslash glsxtremsuffix, \textbackslash glsabbrvemfont and \textbackslash glsfirstabbrvemfont (see §4.5.1.3.9).

\begin{verbatim}
\setabbreviationstyle{short-em-nolong-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert].
\printunsrtglossaries
\end{document}
\end{verbatim}

Example 18: The short-em-nolong-desc abbreviation style


\begin{glossary}
\textbf{SHRT FM (long form)} sample description
\end{glossary}
4. Abbreviations

| short-em-desc | alias: short-em-nolong-desc |

A synonym for short-em-nolong-desc.

| nolong-short-em |

This style is like nolong-short but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont (see §4.5.1.3.9).

```
\setabbreviationstyle{nolong-short-em}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}
```

Example 19: The nolong-short-em abbreviation style


**Glossary**

**SHRT FM** long form

4.5.1.1.2. Long Styles

These styles only show the long form on both first use and subsequent use. See §4.5.1.3.1 and §4.5.1.3.6 for style commands.

| long-noshort-desc |

Only the long form is shown on first use and subsequent use of the \gls-like commands (\glsxtremlongformat). The inline full form uses the same parenthetical style as long-short (\glsxtremlongshortformat). Font variations are available with long-noshort-sc-desc, long-noshort-sm-desc and long-noshort-em-desc.
4. Abbreviations

\setabbreviationstyle{long-noshort-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glsp{ex}[-insert]. \printunsrtglossaries \end{document}

Example 20: The long-noshort-desc abbreviation style


Glossary

long form sample description

The long form is formatted with \glsfirstlongdefaultfont on first use and \glslongdefaultfont for subsequent use and for the \glsxtrlong set of commands. The short form is formatted with \glsfirstabbrvdefaultfont within the inline full form and with \glsabbrvdefaultfont for the \glsxtrshort set of commands. The name is set to the long form (\glsxtrlongnoshortdescname) and the description must be supplied.

long-desc alias: long-noshort-desc

A synonym for long-noshort-desc.

long-noshort

This is like long-noshort-desc but the name is set to the short form (\glsxtrlongnoshortname) and the description is set to the unencapsulated long form.

\setabbreviationstyle{long-noshort}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \gls{ex}[-insert]. \end{document}
Example 21: The long-noshort abbreviation style


Glossary

| SHRT FM | long form |

long

alias: long-noshort

A synonym for long-noshort.

long-noshort-sc

This style is like long-noshort but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).

\setabbreviationstyle{long-noshort-sc}
\newabbreviation{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}
4. Abbreviations

Example 22: The long-noshort-sc abbreviation style


Glossary

SHRT FM long form

Example 23: The long-noshort-sc-desc abbreviation style


Glossary

long-noshort-sc-desc

This style is like long-noshort-desc but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).

\setabbreviationstyle{long-noshort-sc-desc}
\newabbreviation[description={sample description}]{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}-[insert]. Next: \gls{ex}-[insert]. Full: \glsxtrfull{ex}-[insert]. First plural: \glspl{ex}-[insert]. \printunsrtglossaries \end{document}

Example 23: The long-noshort-sc-desc abbreviation style


Glossary

long form sample description

Example 23: The long-noshort-sm abbreviation style


Glossary

long-noshort-sm

This style is like long-noshort but it uses \glsxtrsmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).
4. Abbreviations

\setabbreviationstyle{long-noshort-sm}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

Example 24: The long-noshort-sm abbreviation style


Glossary

SHRT FM long form

\setabbreviationstyle{long-noshort-sm-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

long-noshort-sm-desc

This style is like long-noshort-desc but it uses \glsxtrmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).
4. Abbreviations

Example 25: The long-noshort-sm-desc abbreviation style


Glossary

long form sample description

long-noshort-em

This style is like long-noshort but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont (see §4.5.1.3.9).

\setabbreviationstyle{long-noshort-em}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}

Example 26: The long-noshort-em abbreviation style


Glossary

SHRT FM long form

long-noshort-em-desc

This style is like long-noshort-desc but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont (see §4.5.1.3.9).
Example 27: The `long-noshort-em-desc` abbreviation style


Glossary

\textbf{long form} sample description

This style is like `long-noshort` but it uses \texttt{glsxtremsuffix}, \texttt{glabbrvemfont}, \texttt{glsfirst-abbrvemfont}, \texttt{glslongemfont} and \texttt{glsfirstlongemfont} (see §4.5.1.3.9). This emphasizes both the long and short forms.
4. Abbreviations

Example 28: The long-em-noshort-em abbreviation style


Glossary

SHRT FM long form

long-em-noshort-em-desc

This style is like long-noshort-desc but it uses \glsxtremsuffix, \glsabbrvemfont, \glsfirstabbrvemfont, \glslongemfont and \glsfirstlongemfont (see §4.5.1.3.9). This emphasizes both the long and short forms.

\setabbreviationstyle{long-em-noshort-em-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxstrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}

Example 29: The long-em-noshort-em-desc abbreviation style


Glossary

long form sample description

4.5.1.2. Non-Regular Styles

The following abbreviation styles will set the regular attribute to false if it has previously been set. If it hasn’t already been set, it’s left unset. Other attributes may also be set, depending on the style.
4. Abbreviations

The non-regular styles are too complicated to use \texttt{\glsgenentryfmt} as the display style (with the \texttt{\glslike} commands). Instead they use \texttt{\glsextrgenabbrvfmt}. This means that these styles won't work if you provide your own custom display style (using \texttt{\def\glssentryfmt}) that doesn't check for the \texttt{regular} attribute.

Avoid using \texttt{\glsfirst}, \texttt{\glsfirstplural}, \texttt{\glstext} and \texttt{\glsplural} (or their case-changing variants) with these styles. There are also some styles that can be problematic with \texttt{\GLSname}.

4.5.1.2.1. Long (Short) Styles

These styles show the long form followed by the short form in parentheses on first use. On subsequent use only the short form is shown. See §4.5.1.3.1 and §4.5.1.3.2 for style commands.

The \langle \textit{insert} \rangle is placed after the long form on first use of the \texttt{\glslike} commands and after the short form on subsequent use. The inline full form is the same as the display full form (\texttt{\glsextrlongshortformat}). Font variations are available with \texttt{long-short-sc}, \texttt{long-short-sm}, \texttt{long-short-em} and \texttt{long-em-short-em}.

\begin{verbatim}
\setabbreviationstyle{long-short}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}\[-insert\]. Next: \gls{ex}\[-insert\].
Full: \glsxtrfull{ex}\[-insert\]. First plural:
\glspl[prereset]{ex}\[-insert\].
\printunsrtglossaries
\end{document}
\end{verbatim}

Example 30: The \texttt{long-short} abbreviation style


Glossary

\textbf{SHRT FM} long form

The long form is formatted with \texttt{\glsfirstlongdefaultfont} within the full form and with \texttt{\glslongdefaultfont} for the \texttt{\glsextrlong} set of commands.

The short form is formatted with \texttt{\glsfirstabbrvdefaultfont} within the full form and
4. Abbreviations

with \glsabbrvdefaultfont for subsequent use and for the \glsxtrshort set of commands.

The name is set to the short form (\glsxtrlongshortname) and the description is set to the unencapsulated long form.

As long-short but the description must be supplied in the optional argument of \newabbreviation. Font variations are available with long-short-sc-desc, long-short-sm-desc, long-short-em-desc and long-em-short-em-desc.

\setabbreviationstyle{long-short-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

Example 31: The long-short-desc abbreviation style


Glossary

long form (SHRT FM) sample description

The name and sort are set to the long form followed by the short form in parentheses (\glsxtrlongshortdescname and \glsxtrlongshortdescsort).

This style is like long-short but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).
4. Abbreviations

\setabbreviationstyle{long-short-sc}
\newabbreviation{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}

Example 32: The long-short-sc abbreviation style


Glossary

SHRT FM long form

long-short-sc-desc

This style is like long-short-desc but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).

\setabbreviationstyle{long-short-sc-desc}
\newabbreviation[description={sample description}]{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}
Example 33: The long-short-sc-desc abbreviation style


Glossary

long form (SHRT FM) sample description

long-short-sm

This style is like long-short but it uses \glsxtrsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

\setabbreviationstyle{long-short-sm}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

Example 34: The long-short-sm abbreviation style


Glossary

SHRT FM long form

long-short-sm-desc

This style is like long-short-desc but it uses \glsxtrsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).
4. Abbreviations

\setabbreviationstyle{long-short-sm-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

Example 35: The long-short-sm-desc abbreviation style


Glossary

long form (SHRT FM) sample description

long-short-em

This style is like long-short but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont (see §4.5.1.3.9).

\setabbreviationstyle{long-short-em}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}
4. Abbreviations

Example 36: The long-short-em abbreviation style


Glossary

SHRT FM long form

long-short-em-desc

This style is like long-short-desc but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont (see §4.5.1.3.9).

\setabbreviationstyle{long-short-em-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

Example 37: The long-short-em-desc abbreviation style


Glossary

long form (SHRT FM) sample description

long-em-short-em

This style is like long-short but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont, \glsfirstlongemfont and \glslongemfont (see §4.5.1.3.9). That is, both the long and short forms are emphasized.
4. Abbreviations

\setabbreviationstyle{long-em-short-em}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural:
\glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}

Example 38: The long-em-short-em abbreviation style

form-insert (SHRT FM). First plural: long forms-insert (SHRT FMs).

Glossary

\emph{SHRT FM} long form

long-em-short-em-desc

This style is like long-short-desc but it uses \glsxtrsuffix, \glsabbrvemfont and \glsfirstabbrvemfont, \glsfirstlongemfont and \glslongemfont (see §4.5.1.3.9). That is, both the long and short forms are emphasized.

\setabbreviationstyle{long-em-short-em-desc}
\newabbreviation[description={sample description}]\{ex\}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural:
\glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}
Example 39: The long-em-short-em-desc abbreviation style


Glossary

long form (SHRT FM) sample description

4.5.1.2.2. Long (Short, User) Styles

These styles are like the long (short) styles in §4.5.1.2.1 but additional content can be supplied in the field identified by \glsxtruserfield, which will be placed in the parenthetical content on first use (if set). The inline full form is the same as the display full form.

These styles use the commands \glsxtrusersuffix, \glsabbrvuserfont, \glsfirstabbrvuserfont, \glslonguserfont and \glsfirstlonguserfont (except where noted). See §4.5.1.3.1 and §4.5.1.3.3 for style commands.

If you need to change the font, you can redefine the associated commands (listed above). However, since small caps are awkward because the short plural suffix needs to counteract the small caps, small caps versions are provided.

\setabbreviationstyle{long-short-user}
\newabbreviation[user1={extra info}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert].
\end{document}
4. Abbreviations

Example 40: The long-short-user abbreviation style

Full: long form-insert (SHRT FM, extra info). First plural: long forms-insert (SHRT FMs, extra info).

Glossary

SHRT FM long form

long-short-user-desc

This style is like long-short-user but the description must be supplied. The name is obtained from \glsxtrlongshortuserdescname.

\setabbreviationstyle{long-short-user-desc}
\newabbreviation[description={sample description},
user1={extra info}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}

Example 41: The long-short-user-desc abbreviation style

Full: long form-insert (SHRT FM, extra info). First plural: long forms-insert (SHRT FMs, extra info).

Glossary

long form (SHRT FM, extra info) sample description

This style is incompatible with \GLSname.

If you need to use \GLSname with this style, you’ll have to redefine \glsxtrlongshort-
4. Abbreviations

userdescname so that the field name doesn’t include the entry label. For example:

\newcommand{\glsxtrlongshortuserdescname}{% 
  \protect\glslonguserfont{\the\glslongtok} % 
  \space(\protect\glsabbrvuserfont{\the\glsshorttok}) %
}

long-postshort-user

This style is like long-short-user but the parenthetical material is placed in the post-link hook. Note that, unlike long-short-user, the plural form isn’t used in the parenthetical material. If you require this, you will need to redefine \glsxtrpostusershortformat.

\setabbreviationstyle{long-postshort-user}
\newabbreviation[\useri=\{extra info\}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glspl[prereset]{ex}[-insert]. First plural:
\glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

Example 42: The long-postshort-user abbreviation style


Glossary

SHRT FM long form

long-postshort-user-desc

This style is like long-postshort-user but the description must be supplied. The name is obtained from \glsxtrlongshortuserdescname.
4. Abbreviations

\setabbreviationstyle{long-postshort-user-desc}
\newabbreviation[description={sample description},
    user1={extra info}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}

Example 43: The long-postshort-user-desc abbreviation style

Full: long form-insert (SHRT FM, extra info). First plural: long forms-insert
(SHRT FM, extra info).

Glossary

long form (SHRT FM, extra info) sample description

This style is incompatible with \GLSname. If you need to use \GLSname with this style, you’ll have to redefine \glsxtrshortlongusertdescname so that the field name doesn’t include the entry label, as for long-short-user-desc.

long-postshort-sc-user

This style is like long-postshort-user but it uses \glsxtrscusersuffix, \glsabbrvscuserfont and \glsfirstabbrvscuserfont. The name value is obtained from \glsxtrlongshortscusername.

\setabbreviationstyle{long-postshort-sc-user}
\newabbreviation[user1={extra info}]{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}
4. Abbreviations

Example 44: The `long-postshort-sc-user` abbreviation style


Glossary

```
SHRT FM  long form
```

---

**`long-postshort-sc-user-desc`**

This style is like `long-postshort-sc-user` but the description must be supplied. The name is obtained from `\glsxtrlongshortuserdescname`.

```
\setabbreviationstyle{long-postshort-sc-user-desc}
\newabbreviation[description={sample description},
user1={extra info}]\{ex\}{shrt fm}\{long form\}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prerset]{ex}[-insert].
\end{document}
```

Example 45: The `long-postshort-sc-user-desc` abbreviation style


Glossary

```
long form (SHRT FM, extra info)  sample description
```
4. Abbreviations

This style is incompatible with \GLSname.

If you need to use \GLSname with this style, you’ll have to redefine \glsxtrlongshortscuserdescname so that the field name doesn’t include the entry label.

4.5.1.2.3. Short (Long) Styles

These styles show the short form followed by the long form in parentheses on first use. On subsequent use only the short form is shown. See §4.5.1.3.1 and §4.5.1.3.2 for style commands.

\setabbreviationstyle{short-long}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document}
First: \gls{ex}-\langle insert\rangle. Next: \gls{ex}-\langle insert\rangle. Full: \glsxtrfull{ex}-\langle insert\rangle. First plural: \glspl[prereset]{ex}-\langle insert\rangle.
\end{document}

Example 46: The short-long abbreviation style


Glossary

SHRT FM long form

The long form is formatted with \glsfirstlongdefaultfont within the full form and with \glslongdefaultfont for the \glsxtrlong set of commands.

The short form is formatted with \glsfirstabbrvdefaultfont within the full form and with \glsabbrvdefaultfont for subsequent use and for the \glsxtrshort set of commands.

The name is set to the short form (\glsxtrlongshortname) and the description is set to
the unencapsulated long form.

As short-long but the description must be supplied in the optional argument of \newabbreviation. Font variations are available with short-sc-long-desc, short-sm-long-desc, short-em-long-desc and short-em-long-em-desc.

\setabbreviationstyle{short-long-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

Example 47: The short-long-desc abbreviation style


Glossary

SHRT FM (long form) sample description

The name is set to the short form followed by the long form in parentheses (\glsxtrshortlongdescname), and the sort is set to just the short form (\glsxtrshortlongdescsort).

short-sc-long

This style is like short-long but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).

\setabbreviationstyle{short-sc-long}
\newabbreviation{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural:
4. Abbreviations

Example 48: The short-sc-long abbreviation style

First: \textsc{SHRT FM-insert (long form)}. Next: \textsc{SHRT FM-insert}. Full: \textsc{SHRT FM-insert (long form)}. First plural: \textsc{SHRT FMs-insert (long forms)}.

Glossary

\textsc{SHRT FM} long form

\texttt{short-sc-long-desc}

This style is like \texttt{short-long-desc} but it uses \texttt{glsxtrscsuffix}, \texttt{glabbrvscfont} and \texttt{gls-firstabbrvscfont} (see §4.5.1.3.9).

\begin{verbatim}
\setabbreviationstyle{short-sc-long-desc}
\newabbreviation{description={sample description}}{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glsp{prereset}{ex}[-insert]. \printunsrtglossaries \end{document}
\end{verbatim}

Example 49: The short-sc-long-desc abbreviation style

First: \textsc{SHRT FM-insert (long form)}. Next: \textsc{SHRT FM-insert}. Full: \textsc{SHRT FM-insert (long form)}. First plural: \textsc{SHRT FMs-insert (long forms)}.

Glossary

\textsc{SHRT FM (long form)} sample description
4. Abbreviations

**short-sm-long**

This style is like short-long but it uses \glsxtrmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```latex
\setabbreviationstyle{short-sm-long}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}
```

Example 50: The short-sm-long abbreviation style


**Glossary**

SHRT FM long form

---

**short-sm-long-desc**

This style is like short-long-desc but it uses \glsxtrmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).

```latex
\setabbreviationstyle{short-sm-long-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}
```
4. Abbreviations

Example 51: The short-sm-long-desc abbreviation style


Glossary

SHRT FM (long form) sample description

short-em-long

This style is like short-long but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont (see §4.5.1.3.9).

\setabbreviationstyle{short-em-long}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

Example 52: The short-em-long abbreviation style


Glossary

SHRT FM long form

short-em-long-desc

This style is like short-long-desc but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont (see §4.5.1.3.9).
4. Abbreviations

\setabbreviationstyle{short-em-long-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glsp{ex}[-insert].
\printunsrtglossaries
\end{document}

Example 53: The short-em-long-desc abbreviation style


Glossary

\textit{SHRT FM} (long form) sample description

\setabbreviationstyle{short-em-long-em}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glsp{ex}[-insert].
\printunsrtglossaries
\end{document}

This style is like short-long but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont, \glsfirstlongemfont and \glslongemfont (see §4.5.1.3.9). That is, both the long and short forms are emphasized.
4. Abbreviations

Example 54: The short-em-long-em abbreviation style


Glossary

\textit{SHRT FM long form}

short-em-long-em-desc

This style is like \textit{short-long-desc} but it uses \texttt{\glsxtremsuffix}, \texttt{\glsabbrvemfont} and \texttt{\glsfirstabbrvemfont}, \texttt{\glsfirstlongemfont} and \texttt{\glslongemfont} (see §4.5.1.3.9). That is, both the long and short forms are emphasized.

\setabbreviationstyle{short-em-long-em-desc}
\newabbreviation[description={sample description}] {ex}{SHRT FM}{long form}
\begin{document}
First: \textit{\gls{ex}[-insert]}. Next: \textit{\gls{ex}[-insert]}. Full: \textit{\glsxtrfull{ex}[-insert]}. First plural: \textit{\glspl[prereset]{ex}[-insert]}. \printunsrtglossaries
\end{document}

Example 55: The short-em-long-em-desc abbreviation style


Glossary

\textit{SHRT FM (long form)} sample description

4.5.1.2.4. Short (Long, User) Styles

These styles are like the short (long) styles in §4.5.1.2.3 but additional content can be supplied in the field identified by \texttt{\glsxtruserfield}, which will be placed in the parenthetical
content on first use (if set). The inline full form is the same as the display full form.

These styles use the commands \glsxtrusersuffix, \glsabbrvuserfont, \glsfirstabbrvuserfont, \glslonguserfont and \glsfirstlonguserfont (except where noted). See §4.5.1.3.1 and §4.5.1.3.3 for style commands.

`short-long-user`

This style is like `short-long` but it includes the additional content in the parentheses on first use or the inline full form.

The description is obtained from \glsuserdescription, which can be redefined to include the additional information, if required.

\setabbreviationstyle{short-long-user}
\newabbreviation[\user1={extra info}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[\prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}

Example 56: The short-long-user abbreviation style


Glossary

SHRT FM long form

`short-long-user-desc`

This style is like `short-long-user` but the description must be provided. The name is obtained from \glsxtrshortlonguserdescname and the sort value is obtained from \glsxtrshortlongdescsort.
4. Abbreviations

\setabbreviationstyle{short-long-user-desc}
\newabbreviation[description={sample description},
user1={extra info}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural:
\glsp{ex}[-insert].
\printunsrtglossaries
\end{document}

Example 57: The short-long-user-desc abbreviation style

Full: SHRT FM-insert (long form, extra info). First plural: SHRT FM-
insert (long forms, extra info).

Glossary

SHRT FM (long form, extra info) sample description

This style is incompatible with \GLSname.

If you need to use \GLSname with this style, you’ll have to redefine \glsxtrshortlong-
userdescname so that the field name doesn’t include the entry label. For example:

\newcommand{\glsxtrlongshortuserdescname}{%
  \protect\glsabbrvuserfont{\the\glsshorttok}%
  \space(\protect\glslonguserfont{\the\glslongtok})%
}

This style is like short-long but it includes the additional content in the parentheses on first
use or the inline full form. The parenthetical content is placed in the post-link hook which
can avoid overly long hyperlinks.

The description is obtained from \glsuserdescription, which can be redefined to in-
clude the additional information, if required.
Example 58: The short-postlong-user abbreviation style


Glossary

SHRT FM long form

This style is like short-postlong-user but the description must be provided. The name is obtained from \glsxtrshortlonguserdescname. The sort value is the short form.
4. Abbreviations

Example 59: The short-postlong-user-desc abbreviation style


Glossary

SHRT FM (long form, extra info) sample description

This style is incompatible with \GLSname.

If you need to use \GLSname with this style, you’ll have to redefine \glsxtrshortlonguserdescname so that the field name doesn’t include the entry label, as for short-long-user-desc.

4.5.1.2.5. Hyphen Styles

These styles test if the inserted material start with a hyphen. See §4.5.1.3.1, §4.5.1.3.2 and §4.5.1.3.7 for style commands.

These styles are designed to be used with the markwords attribute and (if the short form has spaces) the markshortwords attribute. If the inserted material starts with a hyphen, the spaces will be replaced with hyphens. This replacement won’t take place if the corresponding attribute wasn’t used to mark the inter-word spaces.

Note that \glsxtrshort and \glsxtrlong (and their plural and case-changing variants) don’t perform the inter-word space substitution. The inline full form is slightly different from the display full form for the “post” styles.

long-hyphen-short-hyphen

This style is like long-short but checks the inserted material for a leading hyphen. The description is the long form encapsulated with \glslonghyphenfont. The name is obtained from \glsxtrlongshortname, and the sort value is obtained from \glsxtrlonghyphenshortsort. The inline full form is the same as the display full form.
4. Abbreviations

Example 60: The long-hyphen-short-hyphen abbreviation style


Glossary

SHRT FM long form

long-hyphen-postshort-hyphen

This style is like long-hyphen-short-hyphen but places the insert and parenthetical material in the post-link hook for the display full form.
4. Abbreviations

Example 61: The long-hyphen-postshort-hyphen abbreviation style


Glossary

SHRT FM  long form

Note that the inline full form (\glsxtrfull) doesn’t show the insert in the post-link hook, but instead places it at the end of the link text. This is because only the \gls-like commands (not the \glstext-like commands) set the placeholder \glsinsert to the supplied insert. If you want the insert to show in the parenthetical part of the post-link hook for the inline full form you need to redefine \glsxtrfullsaveinsert:

\renewcommand*{\glsxtrfullsaveinsert}[2]{\def\glsinsert{#2}}

long-hyphen-short-hyphen-desc

This style is like long-hyphen-short-hyphen but the description must be supplied. The name is obtained from \glsxtrlongshortdescname, and the sort value is obtained from \glsxtrlongshortdescsort.

\setabbreviationstyle{long-hyphen-short-hyphen-desc}
\glssetcategoryattributes{abbreviation}{markwords,markshortwords}{true}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}
4. Abbreviations

Example 62: The long-hyphen-short-hyphen-desc abbreviation style


Glossary

long form (SHRT FM) sample description

long-hyphen-postshort-hyphen-desc

This style is like long-hyphen-short-hyphen-desc but places the insert and parenthetical material in the post-link hook for the display full form.

\setabbreviationstyle{long-hyphen-postshort-hyphen-desc}
\glssetcategoryattributes{abbreviation}{markwords,markshortwords}{true}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \end{document}

Example 63: The long-hyphen-postshort-hyphen-desc abbreviation style


Glossary

long form (SHRT FM) sample description

Note that as with the long-hyphen-postshort-hyphen style, the insert isn’t included in the post-link hook by default for the inline full form. If you want the insert to show in the
4. Abbreviations

post-link hook for the inline full form you need to redefine \glsxtrfullsaveinsert.

\begin{description}[labelsep=0pt]
\item[long-hyphen-noshort-desc-noreg]
This style is like long-noshort-desc-noreg but checks the inserted material for a leading hyphen. The description must be supplied. The name is obtained from \glsxtrlongnoshortdescname, and the sort value is obtained from \glsxtrlonghyphennoshortdescsort.
\end{description}

\begin{verbatim}
\setabbreviationstyle{long-hyphen-noshort-desc-noreg}
\glssetcategoryattributes{abbreviation}{markwords,markshortwords}{true}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}
\end{verbatim}

Example 64: The long-hyphen-noshort-desc-noreg abbreviation style


Glossary

long form sample description

\begin{description}[labelsep=0pt]
\item[long-hyphen-noshort-noreg]
This style is like long-noshort-noreg but checks the inserted material for a leading hyphen. The description is set to the unencapsulated long form. The name is obtained from \glsxtrlongnoshortname, and the sort value is obtained from \glsxtrlonghyphennoshortsort.
\end{description}

\begin{verbatim}
\setabbreviationstyle{long-hyphen-noshort-noreg}
\glssetcategoryattributes{abbreviation}{markwords,markshortwords}{true}
\newabbreviation{ex}{SHRT FM}{long form}
\end{verbatim}
4. Abbreviations

Example 65: The long-hyphen-noshort-noreg abbreviation style


Glossary

SHRT FM  long form

This style is like short-long but checks the inserted material for a leading hyphen. The description is the long form encapsulated with \glslonghyphenfont. The name is obtained from \glsxtrshortlongname and the sort value is obtained from \glsxtrshorthyphen-longsort.
4. Abbreviations

Example 66: The short-hyphen-long-hyphen abbreviation style


Glossary

SHRT FM long form

Example 67: The short-hyphen-postlong-hyphen abbreviation style


Glossary

SHRT FM long form

Note that as with the long-hyphen-postshort-hyphen style, the insert isn’t included in the post-link hook by default for the inline full form. If you want the insert to show in the post-link hook for the inline full form you need to redefine \glsxtrfullsaveinsert (as
described above, for the long-hyphen-postshort-hyphen style).

**short-hyphen-long-hyphen-desc**

This style is like short-hyphen-long-hyphen but the description must be supplied. The name is obtained from `\glsxtrshortlongdescname`, and the sort is obtained from `\glsxtrshortlongdescsort`.

```
\setabbreviationstyle{short-hyphen-long-hyphen-desc}
\glssetcategoryattributes{abbreviation}{markwords,markshortwords}{true}
\newabbreviation[description={sample description}]
{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glsxpl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}
```

Example 68: The short-hyphen-long-hyphen-desc abbreviation style


**Glossary**

**SHRT FM** (long form) sample description

**short-hyphen-postlong-hyphen-desc**

This style is like short-hyphen-long-hyphen-desc but the insert and parenthetical material are placed in the post-link hook for the display full form.

```
\setabbreviationstyle{short-hyphen-postlong-hyphen-desc}
\glssetcategoryattributes{abbreviation}{markwords,markshortwords}{true}
\newabbreviation[description={sample description}]
```

4. Abbreviations

Example 69: The short-hyphen-postlong-hyphen-desc abbreviation style


Glossary

SHRT FM (long form) sample description

Note that as with the long-hyphen-postshort-hyphen style, the insert isn’t included in the post-link hook by default for the inline full form. If you want the insert to show in the post-link hook for the inline full form you need to redefine \glsxtrfullsaveinsert (as described above, for the long-hyphen-postshort-hyphen style).

4.5.1.2.6. Only Styles

These styles only show the long form on first use and only show the short form on subsequent use. The inline full form is the same as the display full form. See §4.5.1.3.1, §4.5.1.3.2 and §4.5.1.3.8 for style commands.

The inline full form uses a parenthetical style with the long form followed by the short form in parentheses.

\setabbreviationstyle{long-only-short-only}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries

The name is obtained from \glsxtronlyname and the sort value is just the short form. The description is the long form encapsulated with \glslongonlyfont.
4. Abbreviations

Example 70: The long-only-short-only abbreviation style


Glossary

SHRT FM  long form

long-only-short-only-desc

This is like long-only-short-only but the description must be supplied. The name is obtained from \glsxtronlydescname and the sort is obtained from \glsxtronlydescsort.

\setabbreviationstyle{long-only-short-only-desc}
\newabbreviation[description={sample description}] {ex}{SHRT FM}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert]. \printunsrtglossaries \end{document}

Example 71: The long-only-short-only-desc abbreviation style


Glossary

long form  sample description

long-only-short-sc-only

This is like long-only-short-only but uses small caps. The name is obtained from \glsxtr-
4. Abbreviations

sconlyname, and it uses \glsabbrvsconlyfont, \glsfirstabbrvsconlyfont and \glsxtrsconlysuffix for the abbreviation fonts and plural suffix.

\setabbreviationstyle{long-only-short-sc-only}
\newabbreviation{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}

Example 72: The long-only-short-sc-only abbreviation style


Glossary

SHRT FM long form

long-only-short-sc-only-desc

This is like long-only-short-only-desc but uses small caps. The name is obtained from \glsxtrsconlydescname, and the sort is obtained from \glsxtrsconlydescsort.

\setabbreviationstyle{long-only-short-sc-only-desc}
\newabbreviation[description={sample description}] {ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert].
Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}
Example 73: The long-only-short-sc-only-desc abbreviation style


Glossary

long form sample description

4.5.1.2.7. Footnote Styles

These styles show the short form (\glsxtrshortformat) with the long form as a footnote on first use. On subsequent use only the short form is shown. See §4.5.1.3.1 and §4.5.1.3.4 for style commands.

The inline full form uses the same parenthetical style as short-long (\glsxtrshortlongformat). Font variations are available with short-sc-footnote, short-sm-footnote and short-em-footnote.

short-footnote

The (insert) is placed after the short form on first use and subsequent use of the \gls-like commands.
The long form is formatted with \glsfirstlongfootnotefont within the full form and with \glslongfootnotefont for the \glsxtrlong set of commands.

The short form is formatted with \glsfirstabbrvdefaultfont within the full form and with \glsabbrvdefaultfont for subsequent use and for the \glsxtrshort set of commands.

The name is set to the short form (\glsxtrfootnotename) and the description is set to the unencapsulated long form.

This style automatically sets the nohyperfirst attribute to true for the entry’s category.

A synonym for short-footnote.

As short-footnote but the description must be supplied in the optional argument of \newabbreviation.
4. Abbreviations

The name is set to the short form followed by the long form in parentheses ($\text{glsxtrfootnotedesc}name$), and the sort is set to just the short form ($\text{glsxtrfootnotedesc}sort$).

footnote-desc

alias: short-footnote-desc

A synonym for short-footnote-desc.

short-postfootnote

Similar to short-footnote but the footnote command is placed in the post-link hook. This avoids the problem of nested hyperlinks caused by the footnote marker hyperlink being inside the link text (which is why the short-footnote style switches on the nohyperfirst attribute). Using the post-link hook makes it possible to check for following punctuation so that the footnote marker can be shifted after the punctuation character.
4. Abbreviations

\setabbreviationstyle{short-postfootnote}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document}

First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries
\end{document}

Example 76: The short-postfootnote abbreviation style

First: SHRT FM-insert.\textsuperscript{1} Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FM-insert.\textsuperscript{2}

\section*{Glossary}

\textbf{SHRT FM} long form

\textsuperscript{1}long form
\textsuperscript{2}long form

\begin{itemize}
  \item \textbf{postfootnote} \emph{alias: short-postfootnote}
  \end{itemize}

A synonym for \textbf{short-postfootnote}.

\begin{itemize}
  \item \textbf{short-postfootnote-desc}
  \end{itemize}

Similar to \textbf{short-footnote-desc} but the footnote command is placed in the post-link hook as with \textbf{short-postfootnote}. 
4. Abbreviations

\setabbreviationstyle{short-postfootnote-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document}
First: \gls{ex}-insert. Next: \gls{ex}-insert. Full: \glsxtrfull{ex}-insert. First plural: \glspl{ex}-insert.
\printunsrtglossaries
\end{document}

Example 77: The abbreviation style short-postfootnote-desc


Glossary
SHRT FM (long form) sample description

\begin{center}
\begin{tabular}{ll}
\texttt{postfootnote-desc} & \textit{alias: short-postfootnote-desc} \\
\end{tabular}
\end{center}

A synonym for \texttt{short-postfootnote-desc}.

\begin{center}
\begin{tabular}{ll}
\texttt{short-sc-footnote} \\
\end{tabular}
\end{center}

This style is like \texttt{short-footnote} but it uses \texttt{\glsxtrscsuffix}, \texttt{\glsabbrvscfont} and \texttt{\glsfirstabbrvscfont} (see §4.5.1.3.9).
4. Abbreviations

\setabbreviationstyle{short-sc-footnote}
\newabbreviation{ex}{shrt fm}{long form}
\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl[prereset]{ex}[-insert]. \printunsrtglossaries \end{document}

Example 78: The short-sc-footnote abbreviation style

First: SHRT FM-insert\textsuperscript{1}. Next: SHRT FM-insert. Full: SHRT FM-insert (long form). First plural: SHRT FMs-insert\textsuperscript{2}.

Glossary

SHRT FM long form

\textsuperscript{1}Long form
\textsuperscript{2}Long forms

This style is like short-footnote-desc but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).
Example 79: The\r
\setabbreviationstyle{short-sc-footnote-desc}\r\newabbreviation[description=\r{s}\short fm\{long form\}\r\begin{document} First:\r\gls{ex}[-insert]. Next: \r\gls{ex}[-insert]. Full: \r\glsxtrfull{ex}[-insert]. First\r\plural:\r\glspl[\prereset]{ex}[-insert]. \r\printunsrtglossaries\r\end{document}
4. Abbreviations

Example 80: The short-sc-postfootnote abbreviation style


Glossary

SHRT FM long form

This style is like short-postfootnote-desc but it uses \glsxtrscsuffix, \glsabbrvscfont and \glsfirstabbrvscfont (see §4.5.1.3.9).
4. Abbreviations

Example 81: The short-sc-postfootnote-desc abbreviation style

\begin{document}
\end{document}

Glossary

SHRT FM (long form) sample description

---

This style is like short-footnote but it uses glsxtrsmsuffix, glsabbrvsmfont and glsfirstabbrvsmfont (see §4.5.1.3.9).
Example 82: The short-sm-footnote abbreviation style


Glossary

SHRT FM long form

\begin{document}
\setabbreviationstyle{short-sm-footnote}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First:
\gls{ex}[-insert]. Next:
\gls{ex}[-insert]. Full:
\glsxtrfull{ex}[-insert]. First plural:
\glspl[prereset]{ex}[-insert].
\printunsrtglossaries
\end{document}

\section*{short-sm-footnote-desc}

This style is like short-footnote-desc but it uses \glsxtrmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).
Example 83: The
abbreviation style

\begin{document}
First: \textsc{SHRT FM}-insert. Next: \textsc{SHRT FM}-insert. Full: \textsc{SHRT FM}-insert (long form). First plural: \textsc{SHRT FM}-inserts.
\end{document}

\noindent
\begin{tabular}{l}
\textbf{Glossary} \\
\textsc{SHRT FM (long form)} sample description
\end{tabular}

\begin{itemize}
\item long form
\item long forms
\end{itemize}

This style is like \texttt{short-postfootnote} but it uses \texttt{\textbackslash glsxtrmsuffix}, \texttt{\textbackslash glsabbrvsmfont} and \texttt{\textbackslash glsfirstabbrvsmfont} (see §4.5.1.3.9).
4. Abbreviations

Example 84: The short-sm-postfootnote abbreviation style


Glossary

SHRT FM long form

This style is like short-postfootnote-desc but it uses \glstextrmsuffix, \glsabbrvsmfont and \glsfirstabbrvsmfont (see §4.5.1.3.9).
Example 85: The short-sm-postfootnote-desc abbreviation style


Glossary

SHRT FM (long form) sample description

This style is like short-footnote but it uses \glsxtremsuffix, \glsabbrvemfont and \glsfirstabbrvemfont (see §4.5.1.3.9).
4. Abbreviations

Example 86: The short-em-footnote abbreviation style


Glossary

\textit{SHRT FM} long form

---

\texttt{\setabbreviationstyle{short-em-footnote}}
\texttt{\newabbreviation{ex}{SHRT FM}{long form}}
\texttt{\begin{document} First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{prereset}{ex}[-insert]. \printunsrtglossaries \end{document}}

This style is like short-footnote-desc but it uses \texttt{\glsxtremsuffix}, \texttt{\glsabbrvemfont} and \texttt{\glsfirstabbrvemfont} (see §4.5.1.3.9).
4. Abbreviations

\setabbreviationstyle{short-em-footnote-desc}
\newabbreviation[description={sample description}]{ex}{SHRT FM}{long form}
\begin{document}
First: \gls{ex}[-insert]. Next: \gls{ex}[-insert]. Full: \glsxtrfull{ex}[-insert]. First plural: \glspl{ex}[-insert].
\printunsrtglossaries
\end{document}

Example 87: The short-em-footnote-desc abbreviation style

First: \textit{SHRT FM-insert}\textsuperscript{1}. Next: \textit{SHRT FM-insert}. Full: \textit{SHRT FM-insert (long form)}. First plural: \textit{SHRT FMs-insert}\textsuperscript{2}.

Glossary

\textit{SHRT FM} (long form) sample description

\begin{itemize}
  \item \textit{long form}
  \item \textit{long forms}
\end{itemize}

short-em-postfootnote

This style is like \textit{short-postfootnote} but it uses \texttt{\glsxtremsuffix}, \texttt{\glsabbrvemfont} and \texttt{\glsfirstabbrvemfont} (see §4.5.1.3.9).
Abbreviations

Example 88: The \texttt{short-em-postfootnote} abbreviation style

\begin{document}
\setabbreviationstyle{short-em-postfootnote}
\newabbreviation{ex}{SHRT FM}{long form}
\begin{document} First: $\gls{ex}$\[-insert\]. Next: $\gls{ex}$\[-insert\]. Full: $\glsxtrfull{ex}$\[-insert\]. First plural: $\glspl[prereset]{ex}$\[-insert\]. \printunsrtglossaries \end{document}

Glossary

\textit{SHRT FM} long form

\texttt{short-em-postfootnote-desc}

This style is like \texttt{short-postfootnote-desc} but it uses \texttt{glsxtremsuffix}, \texttt{glsabbrvemfont} and \texttt{glsfirstabbrvemfont} (see §4.1.3.9).
4. Abbreviations

Example 89: The abbreviation style

\setabbreviationstyle{short-em-postfootnote-desc}
\newabbreviation[description=sample description]{ex}{SHRT FM}{long form}
\begin{document}
First: \gls{ex}. Next: \gls{ex}. Full: \glsxtrfull{ex}. First plural: \glspl{ex}. \printunsrtglossaries \end{document}

4.5.1.2.8. Short Styles

These styles only show the short form on both first use and subsequent use. See §4.5.1.3.1 and §4.5.1.3.5 for style commands. They are essentially identical to the corresponding regular style listed in §4.5.1.2.8 except that they change the \texttt{regular} attribute to false.

\begin{itemize}
  \item \texttt{short-nolong-noreg}
    This style is a non-regular version of the \texttt{short-nolong} style.
  \item \texttt{short-nolong-desc-noreg}
    This style is a non-regular version of the \texttt{short-nolong-desc} style.
  \item \texttt{nolong-short-noreg}
    This style is a non-regular version of the \texttt{nolong-short} style.
\end{itemize}

4.5.1.2.9. Long Styles

These styles only show the long form on both first use and subsequent use. See §4.5.1.3.1 and §4.5.1.3.6 for style commands. They are essentially identical to the corresponding regular
4. Abbreviations

style listed in §4.5.1.2.9 except that they change the regular attribute to false.

```
long-noshort-desc-noreg
```

This style is a non-regular version of the long-noshort-desc style.

```
long-noshort-noreg
```

This style is a non-regular version of the long-noshort style.

```
long-em-noshort-em-noreg
```

This style is a non-regular version of the long-em-noshort-em style.

```
long-em-noshort-em-desc-noreg
```

This style is a non-regular version of the long-em-noshort-em-desc style.

4.5.1.3. Formatting Commands and Hooks

These commands are used by the predefined abbreviation styles. These are considered user commands, which you can redefine to customize the style.

4.5.1.3.1. General

These commands apply to all styles.

```
\ifglsxtrinsertinside (true)\else (false)\fi  initial: \iffalse
```

This conditional is used to determine whether the insert part should go inside or outside of the style’s font formatting commands. The default setting is false.

```
\glsxtrinsertinsidetrue
```

Set the insert inside conditional to true.

```
\glsxtrinsertinsidefalse
```

131
4. Abbreviations

Set the insert inside conditional to false.

\glsxtrparen{(text)}

Used for parenthetical content in the inline full form and also, for some styles, the display full form. Note that this formats the opening and closing parentheses according to the inner formatting, but not the argument, which should already incorporate it. The default definition is:

\newcommand*{\glsxtrparen}[1]{\glsxtrgenentrytextfmt{(}#1\glsxtrgenentrytextfmt{)}}

\glsxtrfullsep{⟨entry-label⟩}

Separator placed before \glsxtrparen. This is a space by default, but it includes the inner formatting. The argument (the entry label) is ignored by default:

\newcommand*{\glsxtrfullsep}[1]{\glsxtrgenentrytextfmt{ }}

You can redefine this to use \glsabsspace if you want to have a non-breakable space if the short form is less than \glsacspacemax in width. (You can use \glsacspace instead, but note that \glsacspace doesn’t incorporate the inner formatting.)

\glsabbrvdefaultfont{(text)}

Abbreviation font command used by styles that don’t have specific font markup (for example, long-short but not long-em-short-em). This just does its argument.

\glsfirstabbrvdefaultfont{(text)}

First use abbreviation font command used by styles that don’t have specific font markup. This is defined to just use \glsabbrvdefaultfont.

\glsxtrdefaultrevert{(text)}
4. Abbreviations

This is the default definition of \glsxtrrevert used by styles that don’t have specific font markup. If you redefine \glsabbrvdefaultfont, you will need to redefine \glsxtrdefaultrevert as applicable.

\texttt{\glslongdefaultfont{⟨text⟩}}

Long form font command used by styles that don’t have specific font markup. This just does its argument.

\texttt{\glsfirstlongdefaultfont{⟨text⟩}}

First use long form font command used by styles that don’t have specific font markup. This is defined to just use \glslongdefaultfont.

4.5.1.3.2. Parenthetical Styles

These commands apply to the parenthetical styles, such as \textit{long–short}.

\texttt{\glsxtrlongshortname}

This command should expand to the value of the name key for styles like \textit{long–short}. The default definition is:

\texttt{\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}}

\texttt{\glsxtrlongshortdescsort}

This command should expand to the sort value used by styles such as \textit{long–short–desc}. The default definition is:

\texttt{\expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)}

Note that this uses the original ⟨long⟩ and ⟨short⟩ values supplied to \newabbreviation.

This command is irrelevant with the “unsrt” family of commands.
4. Abbreviations

\glsxtrlongshortdescname

This command should expand to the name value used by styles such as long-short-desc. The default definition is:

\glsxplongfont{\the\glslongtok}{\glscategorylabel}\%
\protect\glsxtrfullsep{\the\glslabeltok}\%
\protect\glsxtrparen
{\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}}

This essentially expands to ⟨long⟩ ⟨short⟩ but includes the style font changing commands, the inner text formatting, and accessibility support.

\glsxtrshortlongname

This command should expand to the value of the name key for styles like short-long. The default definition is:

\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}

\glsxtrshortlongdescsort

This command should expand to the value of the sort key for styles like short-long-desc. The default definition is just \expandonce\glsxtrorgshort.

This command is irrelevant with the “unsrt” family of commands.

\glsxtrshortlongdescname

This command should expand to the value of the name key for styles like short-long-desc. The default definition is:
4. Abbreviations

4.5.1.3.3. User Styles

These commands apply to the “user” styles, such as long-short-user.

`\glsxtruserfield`

This command should expand to the internal label of the field used to store the additional information that should be shown in the parenthetical material on first use. The default is useri, which corresponds to the user1 key.

`\glsxtruserparensep`

The separator used within the parenthetical content. This defaults to a comma followed by a space.

`\glsxtruserfieldfmt{⟨text⟩}`

Used to encapsulate the value of the field given by `\glsxtruserfield` within `\glsxtruserparen` and `\GLSxtruserparen`. This simply does its argument by default.

The inner formatting with both `\glsxtruserparen` and `\GLSxtruserparen`, and the case-change with the latter, will be included in the argument of `\glsxtruserfieldfmt`.

For example, to emphasize the user value and separate it with a semi-colon instead of a comma:

```latex
\renewcommand{\glsxtruserparensep}{; }
\renewcommand{\glsxtruserfieldfmt}[1]{\emph{#1}}
```

`\glsabbrvuserfont{⟨text⟩}`
4. Abbreviations

Formatting for the “user” short form. This defaults to \glsabbrvdefaultfont.

\glsfirstabbrvuserfont{⟨text⟩}

Formatting for the “user” short form shown on first use. This defaults to \glsabbrvuserfont.

\glsxtrusersuffix  \textit{initial}: \glsxtrabbrvpluralsuffix

Short plural suffix used by the “user” styles. This defaults to \glsxtrabbrvpluralsuffix.

\glslonguserfont{⟨text⟩}

Formatting for the “user” long form. This defaults to \glsabbrvdefaultfont.

\glsfirstlonguserfont{⟨text⟩}

Formatting for the “user” short form shown on first use. This defaults to \glslonguserfont.

\glsabbrvscuserfont{⟨text⟩}

Formatting for the “sc-user” short form. This uses \glsabbrvscfont, which in turn uses \textsc to apply a small caps style, so your document font needs to support it.

\textsc uses small capital glyphs for lowercase characters. Uppercase characters show as normal capitals. This means that you need to use lowercase characters in the abbreviation.

\glsfirstabbrvscuserfont{⟨text⟩}

Formatting for the “sc-user” short form shown on first use. This defaults to \glsabbrvscuserfont.

\glsxtrscuserrevert{⟨text⟩}

136
Counteracts the effect of \glsabbrvscuserfont. The default is \glsxtrscuserrevert. If you redefine \glsabbrvscuserfont, you will need to redefine \glsxtrscuserrevert as applicable.

\glsxtrscusersuffix

Short plural suffix used by the “sc-user” styles. This defaults to \glsxtrscsuffix.

\glsuserdescription{⟨text⟩}{⟨entry-label⟩}

The description field is set to this, where the ⟨text⟩ argument is the long form, for the “user” styles where the description is set by default. This is defined to ignore its second argument:

\newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}

If you want to include the information contained in the field identified by \glsxtruserfield, the second argument provides a way of accessing that field without relying on the \glscurrententrylabel placeholder. For example:

\renewcommand*{\glsuserdescription}[2]{% \glslonguserfont{#1}% \ifglshasfield{\glsxtruserfield}{#2}% {, \glscurrentfieldvalue}% {}% }

\glsxtruserparen{⟨text⟩}{⟨entry-label⟩}

If the field given by \glsxtruserfield has been set, this essentially does:

\glsxtrfullsep{⟨entry-label⟩}\glsxtrparen{⟨text⟩, ⟨user-value⟩}

otherwise it does:

\glsxtrfullsep{⟨entry-label⟩}\glsxtrparen{⟨text⟩}

It’s a little more complicated than this as the definition includes the inner formatting around the comma and the field value (⟨user-value⟩). The comma separator is given by \glsxtr-
4. Abbreviations

userparensep, and the field value is encapsulated with \glsxtruserfieldfmt (with the inner formatting inside).

If you redefine this command, you will also need to redefine the following one in a similar manner.

\GLSxtruserparen{⟨text⟩}{⟨entry-label⟩}

As above but the content of the field given by \glsxtruserfield is converted to all caps. Note that simply applying an uppercase command to \glsxtruserparen can fail as it can cause the label to be converted to all caps, which is the reason why a separate command to internally perform the case-change is provided.

\glsxtrlongshortuserdescname

Expands to the value for the name key for styles like long–short–user–desc. The default definition is:

\protect\glslonguserfont{⟨the\glslongtok⟩%
\protect\glsxtruserparen{\protect\glsabbrvuserfont{⟨the\glsshorttok⟩}{⟨the\glslabeltok⟩

\glsxtrlongshortscusername

Expands to the value for the name key for styles like long–postshort–sc–user styles where the description is automatically set. The default definition is:

\protect\glsabbrvscuserfont{⟨the\glsshorttok⟩

\glsxtrlongshortscuserdescname

Expands to the value for the name key for styles like long–postshort–sc–user–desc. The default definition is:
4. Abbreviations

```
\protect\glslonguserfont\{\the\glslongtok\}%
\protect\glsxtruserparen
{\protect\glsabbrvscuserfont\{\the\glsshorttok\}\{\the\glslabeltok\}

\glsxtrshortlonguserdescname

Expands to the value for the name key for styles like short-long-user-desc. The default definition is:

```
\protect\glsabbrvuserfont\{\the\glsshorttok\}%
\protect\glsxtruserparen
{\protect\glslonguserfont\{\the\glslongpltok\}%
 {\the\glslabeltok}
```

\glsxtruserlongshortformat\{⟨entry-label⟩}\{⟨insert⟩}\{⟨long-fmt-cs⟩}\{⟨short-fmt-cs⟩\}

This command is used on the first use of \gls or with \glsxtrfull by styles like long-short-user to format the long form followed by the short form (with optional user information) in parentheses. The default definition is:

```
\newcommand*{\glsxtruserlongshortformat}[4]{%
 \glsxtrlongformat{#1}{#2}{#3}%
 \glsxtrusershortformat{#1}{#4}%
 }
```

\GLSxtruserlongshortformat\{⟨entry-label⟩\}{⟨insert⟩}\{⟨long-fmt-cs⟩\}{⟨short-fmt-cs⟩\}

As above but for sentence case.

\GLSXtruserlongshortformat\{⟨entry-label⟩\}{⟨insert⟩}\{⟨long-fmt-cs⟩\}{⟨short-fmt-cs⟩\}
4. Abbreviations

As above but for all caps.

```
\glsxtruserlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
```

This command is used on the first use of \glsp or with \glsxtrfullpl by styles like \texttt{long-short-user} to format the plural long form followed by the plural short form (with optional user information) in parentheses. The default definition is:

```
\newcommand*{\glsxtruserlongshortplformat}[4]{%
  \glsxtrlongplformat{#1}{#2}{#3}%
  \glsxtrusershortplformat{#1}{#4}%
}%
```

As above but for sentence case.

```
\GLSxtruserlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
```

As above but for all caps.

```
\glsxtrusershortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
```

This command is used on the first use of \glsp or with \glsxtrfull by styles like \texttt{short-long-user} to format the short form followed by the long form (with optional user information) in parentheses. The default definition is:

```
\newcommand*{\glsxtrusershortlongformat}[4]{%
  \glsxtrshortformat{#1}{#2}{#3}%
  \glsxtruserlongformat{#1}{#4}%
}%
```
4. Abbreviations

\glsxtrusershortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above but for sentence case.

\GLSxtrusershortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above but for all caps.

\glsxtrusershortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

This command is used on the first use of \glspl or with \glsxtrfullpl by styles like short-long-user to format the plural short form followed by the plural long form (with optional user information) in parentheses. The default definition is:

\newcommand*{\glsxtrusershortlongplformat}[4]{% 
  \glsxtrshortplformat{#1}{#2}{#3}\
  \glsxtruserlongplformat{#1}{#4}
}

\glssxtrusershortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above but for sentence case.

\GLSxtrusershortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above but for all caps.

\glsxtrusershortformat{⟨entry-label⟩}{⟨fmt-cs⟩}

Used to format the singular short form in parentheses (with \glsxtruserparen) on the first use of \gls or \Gls or with \glsxtrfull or \Glsxtrfull for styles like long-short-user.
The default definition is:

\newcommand*{\glsxtrusershortformat}[2]{%\glsxtruserparen{\glsxtrshortformat{#1}{#2}{#1}}}%

\glsxtrusershortplformat{⟨entry-label⟩}{⟨fmt-cs⟩}

As \glsxtrusershortformat but for the first use of \glspl or with \glsxtrfull for styles like long-short-user. This has a similar definition to the above but with \glsxtrshortplformat.

\GLSxtrusershortformat{⟨entry-label⟩}{⟨fmt-cs⟩}

As \glsxtrusershortformat but is used with the all caps \GLS or \GLSxtrfull. This uses \GLSxtruserparen instead of \glsxtruserparen.

\GLSxtrusershortplformat{⟨entry-label⟩}{⟨fmt-cs⟩}

As \glsxtrusershortplformat but is used with the all caps \GLSpl or \GLSxtrfullpl. This uses \GLSxtruserparen instead of \glsxtruserparen.

\glsxtrpostusershortformat{⟨entry-label⟩}{⟨fmt-cs⟩}

Used in the post-link hook to format the short form in parentheses for styles like long-postshort-user. The default definition is:

\newcommand*{\glsxtrpostusershortformat}[2]{%\glsxtrifallcaps\{\GLSxtrshortformat{#1}{#2}\}\{\glsxtrshortformat{#1}{#2}{#1}}}%

Note that this doesn’t check if the plural form was used. If you require this, you will need to redefined this command to include \glsifplural:
4. Abbreviations

\renewcommand*{\glsxtrpostusershortformat}[2]{%  
\glsifplural  
\glsxtrifallcaps  
{\GLSxtrusershortplformat{#1}{#2}}%  
{\glsxtrusershortplformat{#1}{#2}}%  
}%  
\glsxtrifallcaps  
{\GLSxtrusershortformat{#1}{#2}}%  
{\glsxtrusershortformat{#1}{#2}}%  
}%}

\glsxtruserlongformat{(entry-label)}{(fmt-cs)}

Used to format the singular long form in parentheses (with \glsxtruserparen) on the first use of \gls or \Gls or with \glsxtrfull for styles like short-long-user. The default definition is:

\newcommand*{\glsxtruserlongformat}[2]{%  
\glsxtruserparen{\glsxtrlongformat{#1}{}{#2}}{#1}%  
}

\GLSxtruserlongformat{(entry-label)}{(fmt-cs)}

As \glsxtruserlongformat but all caps. This uses \GLSxtruserparen instead of \glsxtruserparen.

\glsxtruserlongplformat{(entry-label)}{(fmt-cs)}

As \glsxtruserlongformat but for the first use of \glspl or with \glsxtrfull for styles like short-long-user. This has a similar definition to \glsxtruserlongformat but with
4. Abbreviations

\texttt{\textbackslash glsxtrlongplformat.}

\texttt{\textbackslash GLSxtruserlongplformat\{\textit{entry-label}\}\{\textit{fmt-cs}\}}

As \texttt{\textbackslash glsxtruserlongplformat} but all caps. This uses \texttt{\textbackslash GLSxtruserparen} instead of \texttt{\textbackslash gls-xtruserparen}.

\texttt{\textbackslash glsxtrpostuserlongformat\{\textit{entry-label}\}\{\textit{fmt-cs}\}}

Used in the post-link hook to format the long form in parentheses for styles like \texttt{short-postlong-user}. The default definition is:

\begin{verbatim}
\newcommand*{\glxtrpostuserlongformat}[2]{% 
  \glsxtrifallcaps{\GLSxtruserlongformat{#1}{#2}}% 
  \glxtruserlongformat{#1}{#2}% 
}
\end{verbatim}

Note that, as with \texttt{\textbackslash glsxtrpostusershortformat}, this doesn’t check if the plural form was used. If you require this, you will need to redefine this command to include \texttt{\textbackslash glsifplural}.

4.5.1.3.4. Footnote Styles

These commands are only used by the footnote styles.

\texttt{\textbackslash glsxtrfootnotename}

This command should expand to the value of the \texttt{name} key. The default definition is:

\begin{verbatim}
\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}
\end{verbatim}

\texttt{\textbackslash glsxtrfootnotedescname}

This command should expand to the value of the \texttt{name} key for styles like \texttt{footnote-desc}. The default definition is:
4. Abbreviations

\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}%
\protect\glsxtrfullsep{\the\glslabeltok}%
\protect\glsxtrparen
{\glsxplongfont{\the\glslongtok}{\glscategorylabel}}%

\glsxtrfootnotedescsort

This command should expand to the value of the sort key for styles like footnote-desc. The default definition is simply \the\glsshorttok.

This command is irrelevant with the “unsrt” family of commands.

\glslongfootnotefont{{\text}}

The formatting command used for the long form in the footnote styles. The default is to simply use \glslongdefaultfont.

\glsxfirstlongfootnotefont{{\text}}

The formatting command used for the first use long form in the footnote styles. The default is to simply use \glslongfootnotefont.

\glsxtrabbrvfootnote{{entry-label}}{{\text}}

The command that produces the footnote. The default definition ignores the first argument:

\newcommand*{\glsxtrabbrvfootnote}[2]{\footnote{#2}}{\footnote{#2}}

\glsxtrfootnotelongformat{{entry-label}}{{fmt-cs}}

This command is used within the footnote to display the long form formatted with ⟨fmt-cs⟩ for the footnote styles on first use of \gls, \Gls and \GLS. The default definition is simply:

145
4. Abbreviations

\newcommand*{\glsxtrfootnotelongformat}{\glsxtrlongformat{#1}{#2}}

For example, if the footnote should start with an uppercase letter then simply redefine this to use \Glsxtrlongformat instead:

\renewcommand*{\glsxtrfootnotelongformat}{\Glsxtrlongformat{#1}{#2}}

\glsxtrfootnotelongplformat{⟨entry-label⟩}{⟨fmt-cs⟩}

This command is used within the footnote to display the plural long form formatted with ⟨fmt-cs⟩ for the footnote styles on first use of \glspl, \Glspl and \GLSpl. The default definition is simply:

\newcommand*{\glsxtrfootnotelongplformat}{\glsxtrlongplformat{#1}{#2}}

\glsxtrpostfootnotelongformat{⟨entry-label⟩}{⟨fmt-cs⟩}

This command is used for the “postfootnote” styles. This is simply defined to do \glsxtrfootnotelongformat. Note that there’s no plural equivalent as the “postfootnote” styles don’t check if the plural command (\glspl etc) was used.

4.5.1.3.5. No-Long Styles

These commands are used by the “nolong” styles.

\glsxtrshortnolongname

This command should expand to the value of the name key for styles like short-nolong. The default definition is:
4. Abbreviations

\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}

\glsxtrshortdescname

This command should expand to the value of the name key for styles like short-nolong-desc. The default definition is:

\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}\%\protect\glsxtrfullsep{\the\glslabeltok}\%\protect\glsxtrparen\{\glsxplongfont{\the\glslongtok}{\glscategorylabel}\}%

4.5.1.3.6. No-Short Styles

These commands are used by the “noshort” styles.

\glsxtrlongnoshortdescname

This command should expand to the value of the name key for styles like long-noshort-desc. The default definition is:

\glsxplongfont{\the\glslongtok}{\glscategorylabel}

\glsxtrlongnoshortname

This command should expand to the value of the name key for styles like long-noshort. The default definition is:

\glsxpabbrvfont{\the\glsshorttok}{\glscategorylabel}
4. Abbreviations

4.5.1.3.7. Hyphen Styles

These are commands used by the “hyphen” styles. They are designed to work with the markwords and markshortwords attributes.

\glsabbrvhyphenfont{⟨text⟩}

The formatting command used for the short form in the hyphen styles. The default is to simply use \glsabbrvdefaultfont.

\glsfirstabbrvhyphenfont{⟨text⟩}

The formatting command used for the short form in the hyphen styles on first use. The default is to simply use \glsabbrvhyphenfont.

\glslonghyphenfont{⟨text⟩}

The formatting command used for the long form in the hyphen styles. The default is to simply use \glslongdefaultfont.

\glsfirstlonghyphenfont{⟨text⟩}

The formatting command used for the long form in the hyphen styles on first use. The default is to simply use \glslonghyphenfont.

\glsxtrhyphensuffix initial: \glsxtrabbrvpluralsuffix

Short plural suffix used by the “hyphen” styles. This defaults to \glsxtrabbrvpluralsuffix.

\glsxtrlonghyphenshortsort

Expands to the sort value for the styles like long-hyphen-short-hyphen. This defaults to the original short value (\glsxtrorgshort). This command is irrelevant with the “unsrt” family of commands.
4. Abbreviations

Expands to the sort value for the styles like \texttt{short-hyphen-long-hyphen}. This defaults to the original short value (\texttt{\glsxtrorgshort}). This command is irrelevant with the “unsrt” family of commands.

\begin{verbatim}
\glsxtrlonghyphennoshortsort
\end{verbatim}

Expands to the sort value for the styles like \texttt{long-hyphen-noshort-noreg}. This defaults to the original short value (\texttt{\glsxtrorgshort}). This command is irrelevant with the “unsrt” family of commands.

\begin{verbatim}
\glsxtrlonghyphennoshortdescsort
\end{verbatim}

Expands to the sort value for the styles like \texttt{long-hyphen-noshort-desc-noreg}. This defaults to the original long value (\texttt{\glsxtrorglong}). This command is irrelevant with the “unsrt” family of commands.

\begin{verbatim}
\glsxtrlonghyphenshort$q\{\langle\text{entry-label}\rangle\}\{\langle\text{long}\rangle\}\{\langle\text{short}\rangle\}\{\langle\text{insert}\rangle\}$
\end{verbatim}

Formats the long and short form for the full or first use \texttt{long-hyphen-short-hyphen} style. This uses \texttt{\glsxtrifhyphenstart} to test if the \texttt{\langle\text{insert}\rangle} starts with a hyphen. If it does, \texttt{\glsxtrwordsep} is locally set to \texttt{\glsxtrwordsephyphen} to replace the inter-word spaces with hyphens. The short form is placed in parentheses with \texttt{\glsxtrparen}, preceded by the \texttt{\glsxtrfullsep} separator. The \texttt{\langle\text{insert}\rangle} is placed after both the long and the short form.

\begin{verbatim}
\GLSxtrlonghyphenshort$q\{\langle\text{entry-label}\rangle\}\{\langle\text{long}\rangle\}\{\langle\text{short}\rangle\}\{\langle\text{insert}\rangle\}$
\end{verbatim}

As above, but the \texttt{\langle\text{insert}\rangle} is converted to all caps. The \texttt{\langle\text{short}\rangle} and \texttt{\langle\text{long}\rangle} arguments should be supplied as all caps. Note that it’s not possible to simply do \texttt{\glsxtrlonghyphenshort} with \texttt{\MakeUppercase{\langle\text{insert}\rangle}} as the argument as this will interfere with the check to determine if \texttt{\langle\text{insert}\rangle} starts with a hyphen.

\begin{verbatim}
\glsxtrlonghyphennoshort$q\{\langle\text{entry-label}\rangle\}\{\langle\text{long}\rangle\}\{\langle\text{insert}\rangle\}$
\end{verbatim}

Formats the long form for the full or first use \texttt{long-hyphen-noshort-desc-noreg} style. This uses \texttt{\glsxtrifhyphenstart} to test if the \texttt{\langle\text{insert}\rangle} starts with a hyphen. If it does, \texttt{\glsxtrwordsep} is locally set to \texttt{\glsxtrwordsephyphen} to replace the inter-word spaces with
4. Abbreviations

hyphens. The ⟨insert⟩ is placed after the long form.

\GLSxtrlonghyphennoshort{⟨entry-label⟩}{⟨long⟩}{⟨insert⟩}

As above but converts ⟨insert⟩ to all caps. The ⟨long⟩ argument should already be in all caps. Note that it’s not possible to simply do \glsxtrlonghyphennoshort with \MakeUppercase {⟨insert⟩} as the argument as this will interfere with the check to determine if ⟨insert⟩ starts with a hyphen.

\glsxtrlonghyphen{⟨entry-label⟩}{⟨long⟩}{⟨insert⟩}

Formats the long form for the full or first use long–hyphen–postshort–hyphen style. This is similar to the above, but the ⟨insert⟩ argument is only used to check if it starts with a hyphen. The actual ⟨insert⟩ is placed in the post-link hook.

\xpglsxtrposthyphenshort

This command is used in the post-link hook for the long–hyphen–postshort–hyphen style on first use. It expands the placeholder commands (\glslabel and \glsinsert) and uses \GLSxtrposthyphenshort for all caps or \glsxtrposthyphenshort otherwise. Note that this doesn’t show the plural by default. If you require the plural form, you need to redefine this to add a check with \glsifplural:

\newcommand*{\xpglsxtrposthyphenshort}{% 
  \glsifplural  
  {%  
    \glsxtrifallcaps  
    {%  
      \expandafter\GLSxtrposthyphenshortpl\expandafter\glslabel  
      \expandafter{\glsinsert}\%  
    }%  
    {%  
      \expandafter\glsxtrposthyphenshortpl\expandafter\glslabel  
      \expandafter{\glsinsert}\%  
    }%  
  }%  
  {%  
    \glsxtrifallcaps  
  }%}
4. Abbreviations

\begin{verbatim}
\glsposthyphenshort\expandafter\glslabel
\expandafter{\glsinsert}%
\}
\end{verbatim}

\begin{verbatim}
\glsxtrposthyphenshort\langle\entry-label\rangle\{\langle\insert\rangle\}
\end{verbatim}

If \langle\insert\rangle starts with a hyphen, \glsxtrwordsep is locally set to \glsxtrwordsephyphen to replace the inter-word spaces with hyphens. The \langle\insert\rangle encapsulated with \glsfirstlonghyphenfont is then done (to complete the long form, which has already been displayed with \glsxtrlonghyphen in the link text). Then the short form followed by the \langle\insert\rangle is placed in parentheses (with \glsxtrparen preceded by \glsxtrfullsep).

\begin{verbatim}
\GLSxtrposthyphenshort\langle\entry-label\rangle\{\langle\insert\rangle\}
\end{verbatim}

As above but all caps.

\begin{verbatim}
\glsxtrposthyphenshortpl\langle\entry-label\rangle\{\langle\insert\rangle\}
\end{verbatim}

As \glsxtrposthyphenshort but plural.

\begin{verbatim}
\GLSxtrposthyphenshortpl\langle\entry-label\rangle\{\langle\insert\rangle\}
\end{verbatim}

As above but all caps.

\begin{verbatim}
\xpglsxtrposthyphensubsequent
\end{verbatim}

This command is used in the post-link hook for the long-hyphen-postshort-hyphen style on subsequent use. It expands the placeholder commands (\glslabel and \glsinsert) and uses \GLSxtrposthyphensubsequent for all caps or \glsxtrposthyphensubsequent.
4. Abbreviations

otherwise.

\glsxtrposthyphensubsequent\{entry-label\}\{\langle insert\rangle\}

This command is used in the post-link hook for the long-hyphen-postshort-hyphen style on subsequent use. Only the \langle insert\rangle is done.

\GLSxtrposthyphensubsequent\{entry-label\}\{\langle insert\rangle\}

As above but all caps.

\glsxtrshorthyphenlong\{entry-label\}\{short\}\{long\}\{\langle insert\rangle\}

Formats the short and long form for the full or first use short-hyphen-long-hyphen style. Similar to \glsxtrlonghyphenshort but the short and long forms are swapped.

\GLSxtrshorthyphenlong\{entry-label\}\{short\}\{long\}\{\langle insert\rangle\}

As above, but the \langle insert\rangle is converted to all caps. The \langle short\rangle and \langle long\rangle arguments should be supplied as all caps. Note that it’s not possible to simply do \glsxtrshorthyphenlong with \MakeUppercase\{\langle insert\rangle\} as the argument as this will interfere with the check to determine if \langle insert\rangle starts with a hyphen.

\glsxtrshorthyphen\{short\}\{entry-label\}\{\langle insert\rangle\}

Formats the short form for the full or first use short-hyphen-postlong-hyphen style. The \langle insert\rangle argument is only used to check if it starts with a hyphen. The actual \langle insert\rangle is placed in the post-link hook.

\xpglsxtrposthyphenlong

This command is used in the post-link hook for the short-hyphen-postlong-hyphen style on first use. It expands the placeholder commands (\glslabel and \glsinsert) and uses \GLSxtrposthyphenlong for all caps or \glsxtrposthyphenlong otherwise. Note that this doesn’t show the plural by default. If you require the plural form, you need to redefine this to add a check with \glsifplural:
This command is used in the post-link hook for the `short-hyphen-postlong-hyphen` style on first use. Similar to \texttt{\glsxtrposthyphenshort} but shows the long form instead of the short form.

As above but all caps.
4. Abbreviations

As \texttt{\textbackslash glsxtrposthyphenlong} but plural.

\texttt{\textbackslash GLSxtrposthyphenlongpl\{\langle entry-label\rangle\}\{\langle insert\rangle\}}

As above but all caps.

4.5.1.3.8. Only Styles

These are commands used by the “only” styles, such as \texttt{\textbackslash long-only-short-only}.

\texttt{\textbackslash glsabbrvonlyfont\{\langle text\rangle\}}

The formatting command used for the short form in the only styles. The default is to simply use \texttt{\textbackslash glsabbrvdefaultfont}.

\texttt{\textbackslash glsfirstabbrvonlyfont\{\langle text\rangle\}}

The formatting command used for the short form in the only styles on first use. The default is to simply use \texttt{\textbackslash glsabbrvonlyfont}.

\texttt{\textbackslash glslongonlyfont\{\langle text\rangle\}}

The formatting command used for the long form in the only styles. The default is to simply use \texttt{\textbackslash glslongdefaultfont}.

\texttt{\textbackslash glsfirstlongonlyfont\{\langle text\rangle\}}

The formatting command used for the long form in the only styles on first use. The default is to simply use \texttt{\textbackslash glslongonlyfont}.

\texttt{\textbackslash glsxtronlysuffix initial: \textbackslash glsxtrabbrvpluralsuffix}

Short plural suffix used by the “only” styles. This defaults to \texttt{\textbackslash glsxtrabbrvpluralsuffix}.

\texttt{\textbackslash glsabbrvsconlyfont\{\langle text\rangle\}}
4. Abbreviations

The formatting command used for the short form in the “sc-only” styles. The default is to simply use \glsabbrvscfont.

```
\glsfirstabbrvsconlyfont{⟨text⟩}
```

The formatting command used for the short form in the “sc-only” styles on first use. The default is to simply use \glsabbrvsconlyfont.

```
\glsxtrsconlyrevert{⟨text⟩}
```

Counteracts the effect of \glsabbrvsconlyfont. The default is \glsxtrscrevert. If you redefine \glsabbrvsconlyfont, you will need to redefine \glsxtrsconlyrevert as applicable.

```
\glsxtrsconlysuffix
```

Short plural suffix used by the “sc-only” styles. This defaults to \glsxtrcsuffix.

```
\glsxtronlyname
```

Expands to the value for the name key for the “only” styles. The default definition is:

```
\protect\glsabbrvonlyfont{\the\glsshorttok}
```

```
\glsxtronlydescname
```

Expands to the value for the name key for the “only” styles where the description should be described, such as long-only-short-only-desc. The default definition is:

```
\protect\glslongfont{\the\glslongtok}
```

```
\glsxtronldescsort
```

155
4. Abbreviations

Expands to the value for the `sort` key for the “only” styles where the description should be described, such as `long-only-short-only-desc`. The default definition is `\the\glslongtok`.

This command is irrelevant with the “unsrt” family of commands.

\texttt{\glsxtrssonlyname}

Expands to the value for the `name` key for the “sc-only” styles. The default definition is:

\texttt{\protect\glsabbrvssonlyfont{\the\glsshorttok}}

\texttt{\glsxtrssonlydescname}

Expands to the value for the `name` key for the “sc-only” styles where the description should be described. The default definition is to simply use `\glsxtrssonlydescname`.

\texttt{\glsxtrssonlydescsort}

Expands to the value for the `sort` key for the “sc-only” styles where the description should be described, such as `long-only-short-only-desc`. The default definition is to simply use `\glsxtrssonlydescsort`.

This command is irrelevant with the “unsrt” family of commands.

4.5.1.3.9. Fonts

These are commands used by styles that use a particular font shape or size, identified by one of the following two-letter tags: “sc” (`\textsc`), “sm” (`\textsmaller`) or “em” (`\emph`).

For the “sc-user” styles, see §4.5.1.3.3. For the “sc-only” styles, see §4.5.1.3.8.

\texttt{\glsabbrvscfont{⟨text⟩}}
4. Abbreviations

Formatting for the “sc” short form. This uses \textsc to apply a small caps style, so your document font needs to support it.

\textsc uses small capital glyphs for lowercase characters. Uppercase characters show as normal capitals. This means that you need to use lowercase characters in the abbreviation.

\glsfirstabbrvscfont{⟨text⟩}

.Formatting for the “sc” short form shown on first use. This defaults to \glsabbrvscfont.

\glsxtrscrevert{⟨text⟩}

Counteracts the effect of \glsabbrvscfont. This defaults to \glstextup. If you redefine \glsabbrvscfont, you will need to redefine \glsxtrscrevert as applicable.

\glsxtrscsuffix

Short plural suffix used by the “sc” styles. This needs to counteract the smallcaps, so it's defined as:

\protect\glstextup\glsxtrabbrvpluralsuffix

\glsabbrvsmfont{⟨text⟩}

.Formatting for the “sm” short form. This uses \textsmaller, which is defined by the relsize package. You will need to load that package if you want to use any of the “sm” styles.

\textsmaller reduces the font size, so if you want to use it to simulate small caps, you need to use uppercase characters in the abbreviation.

\glsfirstabbrvsmfont{⟨text⟩}
4. Abbreviations

Formatting for the “sm” short form shown on first use. This defaults to \glsabbrvsmfont.

\glsxtrsmrevert\{\langle\text\rangle\}

Counteracts the effect of \glsabbrvsmfont. This defaults to \textlarger. If you redefine \glsabbrvsmfont, you will need to redefine \glsxtrsmrevert as applicable.

\glsxtrmsuffix_{initial: \glsxtrabbrvpluralsuffix}

Short plural suffix used by the “sm” styles. This defaults to \glsxtrabbrvpluralsuffix.

\glsabbrvemfont\{\langle\text\rangle\}

Formatting for the “em” short form. This uses \emph.

\glsfirstabbrvemfont\{\langle\text\rangle\}

Formatting for the “em” short form shown on first use. This defaults to \glsabbrvemfont.

\glsxxtremrevert\{\langle\text\rangle\}

Counteracts the effect of \glsabbrvemfont. This defaults to \textup. If you redefine \glsabbrvemfont, you will need to redefine \glsxxtremrevert as applicable.

\glsxtremsuffix_{initial: \glsxtrabbrvpluralsuffix}

Short plural suffix used by the “em” styles. This defaults to \glsxtrabbrvpluralsuffix.

\glslongemfont\{\langle\text\rangle\}

Formatting for the “em” long form. This uses \emph.

\glsfirstlongemfont\{\langle\text\rangle\}

Formatting for the “em” short form shown on first use. This defaults to \glslongemfont.
4. Abbreviations

4.5.2. Advanced Style Commands

These commands should typically not be needed in a document, but are provided for advanced users. See §4.5.1.3 for commands to adjust the predefined abbreviation styles.

\glssetabbrvfmt{(category)}

Sets the current formatting commands (§4.5.3.2) associated with the abbreviation style associated with the given category. That is, the command redefinitions provided in the third argument ((display definitions)) of \newabbreviationstyle are applied.

If no abbreviation style has been set for the given category, the style associated with the abbreviation category is used.

This command is used:

- At the start of \glsentryfmt if the current entry has the short field set. This ensures that the \gls-like commands use the appropriate formatting.
- At the start of \glsxtrassignfieldfont if the current entry has the short field set. This ensures that the \gls-like commands use the appropriate formatting (where possible).
- At the start of \glsxtrshort, \glsxtrlong, \glsxtrfull and their plural and case-changing variants.
- At the start of glossentryname, glossentrynameother, glossentrydesc, Glossentrydesc, glossentrysymbol and Glossentrysymbol.

\glsuseabbrvfont{(style-name)}{(text)}

A robust command that applies the abbreviation font for the given category to the supplied text.

\glsuselongfont{(style-name)}{(text)}

A robust command that applies the long font for the given category to the supplied text.

\GlsXtrUseAbbrStyleSetup{(style-name)}
This implements the given abbreviation style’s setup code. Note that this expects the place-
holder macros and token registers to be set. This may be used in the \(\text{setup}\) of \texttt{\newabbreviationstyle} to inherit the setup code of a related style.

\GLsXtrUseAbbrStyleFmts{\langle style-name\rangle}

This implements the given abbreviation style’s display definitions code. This may be used in
the \(\text{display definitions}\) of \texttt{\newabbreviationstyle} to inherit the formatting of a related
style.

\GlsXtrUseAbbrStyleFmts{\langle style-name\rangle}

This is used by styles like \texttt{postfootnote} to ensure that the label and inner and outer formatting
are expanded before being passed to \texttt{\glsxtrpostabbrvfootnote}, otherwise they may lose
their definitions before the footnote text is typeset.

\xpglsxtrpostabbrvfootnote

This is used by the footnote styles that defer the footnote to the post-link hook. The default
definition is:

\newrobustcmd*{\glsxtrpostabbrvfootnote}\[2\]{{% 

\glsxtrpostabbrvfootnote{#1}%

{#2}\glsxtrpostfootnotelongformat{#1}{\glsfirstlongfootnotefont}}%
}

The second argument will be the expansion of \texttt{\glsxtrassignlinktextfmt}, to allow the
inner formatting to be picked up, if required.

\glsxtrifhyphenstart{\langle text\rangle}{\langle true\rangle}{\langle false\rangle}

This command is used by the hyphen styles to determine if the insert material starts with a
hyphen. Does \langle true\rangle if \langle text\rangle starts with a hyphen otherwise does \langle false\rangle.

\GlsXtrWarn DeprecatedAbbrStyle{\langle old-name\rangle}{\langle new-name\rangle}

This command is used to generate a warning (with \texttt{\GlossariesExtraWarning}) if a depre-
cated abbreviation style is used.
4. Abbreviations

4.5.3. Defining New Abbreviation Styles

If none of the predefined styles suit your requirements, you can define your own custom style using:

\newabbreviationstyle{⟨style-name⟩}{⟨setup⟩}{⟨display definitions⟩}

The first argument is the style name. This is used internally to form control sequences, so the name shouldn’t contain any special characters.

The second argument sets up the information that’s required when an abbreviation is defined (which is why the style must be set before the abbreviations with that style are defined). The relevant commands for this argument are listed in §4.5.3.1.

The third argument defines the commands that determine how the display style (\gls) and the inline style (\glsxtrfull) are formatted. The relevant commands for this argument are listed in §4.5.3.2.

\renewabbreviationstyle{⟨style-name⟩}{⟨setup⟩}{⟨display definitions⟩}

Redefines an existing abbreviation style.

\letabbreviationstyle{⟨new style⟩}{⟨existing style⟩}

Defines a synonym of an existing abbreviation style.

4.5.3.1. Style Initialisation Hooks

The style initialisation hooks should be placed in the second argument (⟨setup⟩) of \newabbreviationstyle. They ensure that all the fields are correctly initialised when the entry is defined with the underlying \newglossaryentry command. They may also be used to set category attributes.

The following is prepended to ⟨setup⟩ to initialise the final hook:

\renewcommand*{\GlsXtrPostNewAbbreviation}{}

When an entry is defined with \newabbreviation, the following steps are performed:

1. Token registers are initialised to the information provided in the arguments of \newabbreviation: \glskeylisttok, \glslabeltok, \glsshorttok and \glslongtok.

2. The commands \glsxtrorgkeylist, \glsxtrorgshort and \glsxtrorglong are defined to the options, short and long values supplied to \newabbreviation. (The \glskeylisttok \glsshorttok and \glslongtok token registers may be changed...
4. Abbreviations

before the entry is actually defined. These commands may be used to obtain the original values.)

3. \ExtraCustomAbbreviationFields is initialised to do nothing.

4. Accessibility settings are initialised, if required. These redefine \ExtraCustomAbbreviationFields to set the accessibility fields.

5. The command \glscategorylabel is defined to abbreviation.

6. The options list is parsed for the following keys: category and, if accessibility is enabled, access, textaccess, pluralaccess, firstaccess, firstpluralaccess, shortaccess, shortpluralaccess, longaccess, and longpluralaccess.

7. The abbreviation style is applied for the category given by \glscategorylabel (which may have been changed when the options were parsed in the previous step) or the fallback if no abbreviation style is associated with that category. This performs both the ⟨setup⟩ and ⟨display definitions⟩ provided when the style was defined with \newabbreviationstyle.

8. The long plural form is initialised to its default value (⟨long⟩\glspluralsuffix).

9. The markwords attribute, if set, is implemented for the singular long form. It will also mark the entry as having a description with formatting (using \glsexclapplyinnerfmtfield).

10. The markshortwords attribute is implemented, if set, otherwise the insertdots attribute is implemented, if set, for the singular short form.

11. The aposplural attribute is implemented, if set, otherwise the noshortplural attribute is implemented, if set. This step will set the default short plural.

12. \glsshorttok is updated to reflect any changes.

13. The \glsxtrnewabbrevpresetkeyhook hook is performed.

14. The options list is parsed for the shortplural and longplural keys. The \glskeylisttok token is updated to only include the remaining keys that haven’t yet been processed.

15. The markwords attribute, if set, is implemented for the plural long form.

16. The markshortwords attribute, if set, otherwise the insertdots attribute, if set, is implemented for the plural short form.

17. The \glsshortpltok and \glslongpltok registers are set.

18. \newabbreviationhook performed.

19. The entry is defined using \newglossaryentry with the key value list:
4. Abbreviations

20. Add the name, first, firstplural, text and plural keys to the list of inner formatting exclusions, as they include formatting commands.

21. Final hook \GlsXtrPostNewAbbreviation performed.

Note that when these hooks (except the last) are used, the entry hasn’t yet been defined. However, some information will have already been picked up from the arguments of \newabbreviation. These can be accessed in the hooks using the following (but make sure they are fully expanded):

\glscategorylabel

Expands to the entry’s category label.

\glskeylisttok

A token register that contains the options that were passed to \newabbreviation with preprocessed options removed. Use \the\glskeylisttok to expand it.

The original option list, as supplied to \newabbreviation, can be obtained with:

\glsxtrorgkeylist

(Not a token register.)
4. Abbreviations

A token register that contains the entry’s label. Use \the\glslabeltok to expand it.

\glsshorttok

A token register that contains the short form (which may have been modified after being passed to \newabbreviation). Use \the\glsshorttok to expand it.

The original short form, as supplied to \newabbreviation, can be obtained with:

\glsxtrorgshort

(Not a token register.)

\glsshortpltok

A token register that contains the short plural form (which may have been obtained from the short form or modified after being passed to \newabbreviation). Use \the\glsshortpltok to expand it.

\glslongtok

A token register that contains the long form (which may have been modified after being passed to \newabbreviation). Use \the\glslongtok to expand it.

The original long form, as supplied to \newabbreviation, can be obtained with:

\glsxtrorglong

(Not a token register.)

\glslongpltok

A token register that contains the long plural form (which may have been obtained from the long form or modified after being passed to \newabbreviation). Use \the\glslongpltok to expand it.

\ExtraCustomAbbreviationFields

Expands to additional field definitions for the entry. This is used to add the accessibility fields (such as shortaccess), if enabled. The abbreviation style may append (\appto) or prepend
4. Abbreviations

\preto additional information, if required, to this hook.

If you alter this hook, make sure that you include the trailing comma after each \(key\) =\{value\}, including the last one.

\customabbreviationfields

Expands to the default field definitions for the entry. Take care to protect any commands that shouldn’t be expanded. The comma may be omitted from the final \(key\) =\{value\}.

\glsxtrpostnewabbreviation

A hook that’s used after the entry has been defined (at the end of \newabbreviation). This can be used to set category attributes, define the post-link hook, or mark the entry as having a complex style (with \glsxtrsetcomplexstyle).

For example, the long-short abbreviation style includes the following in \setup:

\begin{verbatim}
\renewcommand*{\GlsXtrPostNewAbbreviation}{% 
   \glsxtrsetcomplexstyle{\the\glslabeltok}{3}% 
   \glshasattribute{\the\glslabeltok}{regular}% 
   {% 
     \glsxsetattribute{\the\glslabeltok}{regular}{false}% 
   }% 
{}% 
\}
\end{verbatim}

Note that in the above, the commands within the definition of \GlsXtrPostNewAbbreviation are all expanded when that hook is used. However, if this hook defines other commands or hooks that will be used later, then make sure that the definitions of those commands use the inner hook’s own placeholder commands.

Remember that the post-link hook uses \glslabel to reference the current label. Don’t use \glslabeltok as that will contain the label of the last abbreviation to be defined.

For example, the long-hyphen-postshort-hyphen style has:
In the above, \texttt{\textbackslash glslabeltok} and \texttt{\textbackslash glscategorylabel} are used in the parts that will be expanded at the end of \texttt{\textbackslash newabbreviation}, but \texttt{\textbackslash glslabel} and \texttt{\textbackslash glsinsert} are used in the definition of the post-link hook, which won’t be expanded until the entry is referenced in the document with a command such as \texttt{\textbackslash gls}. (The use of \texttt{\expandafter} is included to assist \texttt{innertextformat}.)

\texttt{\textbackslash glsxtrsetcomplexstyle\{⟨entry-label⟩\}{⟨n⟩}}

This command should go in the definition of \texttt{\GlsXtrPostNewAbbreviation} to indicate that the entry given by \texttt{⟨entry-label⟩} has an abbreviation style that is complex. The second argument \texttt{⟨n⟩} should be numeric and indicates why it doesn’t work with \texttt{\glsfirst}, \texttt{\Glsfirst}, \texttt{\gLsfirst}, \texttt{\glfirstplural}, \texttt{\Glsfirstplural} or \texttt{\GLSfirstplural}: 1 (all caps doesn’t work), 2 (all caps and insert doesn’t work), 3 (insert doesn’t work).

\texttt{\glxstrfirstinnerfmtabbrvfont\{⟨text⟩\}}

This is a robust command that applies both \texttt{\glxstrfirstabbrvfont} and the inner formatting
4. Abbreviations

command \glsxtrgenentrytextfmt. This is used by the following command.

\glsfirstxpabbrvfont{(text)}{(category)}

If the markshortwords attribute is true, this does \protect\glsfirstabbrvfont{(text)} otherwise it does \glsfirstinnerfmtabbrvfont{(text)}.

This command is designed to be used within \CustomAbbreviationFields to set the first and firstplural keys, so it needs to partially expand within \newabbreviation. For example, the postfootnote includes the following lines in the definition of \CustomAbbreviationFields:

\begin{verbatim}
first={\glsfirstxpabbrvfont{\the\glsshorttok}{\glscategorylabel}},%
firstplural={\glsfirstxpabbrvfont{\the\glsshortpltok}{\glscategorylabel}},
\end{verbatim}

This will be expanded before being passed to \newglossaryentry. If the markshortwords attribute is true, this will end up as:

\begin{verbatim}
first={\protect\glsfirstabbrvfont{(short)}},%
firstplural={\protect\glsfirstabbrvfont{(shortpl)}}
\end{verbatim}

otherwise it will end up as:

\begin{verbatim}
first={\glsfirstinnerfmtabbrvfont{(short)}},%
firstplural={\glsfirstinnerfmtabbrvfont{(shortpl)}},%
\end{verbatim}

where (short) and (shortpl) are, respectively, the values in the \glsshorttok and \glsshortpltok registers.

The placeholder registers and macros (such as \glsshorttok and \glscategorylabel) must be expanded before being passed to \newglossaryentry as their values are unreliable outside of \newabbreviation.

\glsinnerfmtabbrvfont{(text)}

This is a robust command that applies both \glsabbrvfont and the inner formatting com-
4. Abbreviations

This is used by the following command.

\glsxpabbrvfont{⟨text⟩}{⟨category⟩}

If the \texttt{markshortwords} attribute is true, this does \protect\glsabbrvfont{⟨text⟩} otherwise it does \glsinnerfmtabbrvfont{⟨text⟩}. This command is designed for the \texttt{name}, \texttt{text} and \texttt{plural} keys within \texttt{\CustomAbbreviationFields}.

\glsfirstinnerfmtlongfont{⟨text⟩}

This is a robust command that applies both \glsfirstlongfont and the inner formatting command \glsxtrgenentrytextfmt. This is used by the following command.

\glsfirstxplongfont{⟨category⟩}{⟨text⟩}

If the \texttt{markwords} attribute is true, this does \protect\glsfirstlongfont{⟨text⟩} otherwise it does \glsfirstinnerfmtlongfont{⟨text⟩}. This command is designed for the \texttt{first} and \texttt{firstplural} keys within \texttt{\CustomAbbreviationFields}.

\glsinnerfmtlongfont{⟨text⟩}

This is a robust command that applies both \glslongfont and the inner formatting command \glsxtrgenentrytextfmt. This is used by the following command.

\glsxplongfont{⟨category⟩}{⟨text⟩}

If the \texttt{markwords} attribute is true, this does \protect\glslongfont{⟨text⟩} otherwise it does \glsinnerfmtlongfont{⟨text⟩}. This command is designed for the \texttt{name}, \texttt{text} and \texttt{plural} keys within \texttt{\CustomAbbreviationFields} (if they should include the long form in their value, such as the \texttt{long-noshort-desc} style).

\glsxtrAccSuppAbbrSetNoLongAttrs{⟨category⟩}

If accessibility support has been enabled with \texttt{accsupp}, this command will initialise support for the \texttt{name}, \texttt{first}, \texttt{firstplural}, \texttt{text} and \texttt{plural} fields for the given category (using \texttt{\glsxtrprovideaccsuppcmd}). The \texttt{nameshortaccess}, \texttt{firstshortaccess} and \texttt{textshortaccess} attributes are set to \texttt{true}. (Does nothing if accessibility support has not been enabled.)
This command is provided for abbreviation styles where the name, first and text are just the formatted abbreviation. The first field may just be the long form or may be a combination of short and long.

\glsxtrAccSuppAbbrSetNameLongAttrs{⟨category⟩}

If accessibility support has been enabled with accsupp, this command will initialise support for the first, firstplural, text and plural fields for the given category (using \glsxtrprovideaccsuppcmd). The firstshortaccess and textshortaccess attributes are set to true. (Does nothing if accessibility support has not been enabled.)

This command is provided for abbreviation styles where the first and text are just the formatted abbreviation. The name field may just be the long form or may be a combination of short and long.

\glsxtrAccSuppAbbrSetFirstLongAttrs{⟨category⟩}

If accessibility support has been enabled with accsupp, this command will initialise support for the name, text and plural fields for the given category (using \glsxtrprovideaccsuppcmd). The nameshortaccess and textshortaccess attributes are set to true. (Does nothing if accessibility support has not been enabled.)

This command is provided for abbreviation styles where the name and text are just the formatted abbreviation. The first field may just be the long form or may be a combination of short and long.

\glsxtrAccSuppAbbrSetTextShortAttrs{⟨category⟩}

If accessibility support has been enabled with accsupp, this command will initialise support for the text and plural fields for the given category (using \glsxtrprovideaccsuppcmd). The textshortaccess attribute is set to true. (Does nothing if accessibility support has not been enabled.)

This command is provided for abbreviation styles where the text is just the formatted abbreviation. The name and first fields may just be the long form or may be a combination of short and long. The name may also be short but followed by the long form in the description.

\glsxtrAccSuppAbbrSetNameShortAttrs{⟨category⟩}

If accessibility support has been enabled with accsupp, this command will initialise support for the name field for the given category (using \glsxtrprovideaccsuppcmd). The nameshortaccess attribute is set to true. (Does nothing if accessibility support has not been enabled.)
This command is provided for abbreviation styles where only the name is just the formatted abbreviation. The first and text fields may just be the long form or may be a combination of short and long.

### 4.5.3.2. Style Formatting Commands

The final \textit{display definitions} argument of \texttt{\newabbreviationstyle} should contain the redefinitions of the style commands listed here that are used to format abbreviations.

Whenever an abbreviation style is activated with commands like \texttt{\setabbreviationstyle}, \texttt{\newabbreviation} or \texttt{\glssetabbrvfmt}, \textit{display definitions} are implemented.

If you simply want to adjust the formatting of one of the predefined styles, you should redefine the associated commands listed in §4.5.1.3.

The following initialisation is always prepended to \textit{display definitions} so you can omit them if the default is appropriate for your style:

\begin{verbatim}
\renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}\
\renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}\
\renewcommand*{\GLSxtrinlinefullformat}{\GLSxtrfullformat}\
\renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}\
\renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}\
\renewcommand*{\GLSxtrinlinefullplformat}{\GLSxtrfullplformat}\
\let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt\
\let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt\
\let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt\
\let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt\
\let\GLSxtrsubsequentfmt\GLSxtrdefaultsubsequentfmt\
\let\GLSxtrsubsequentplfmt\GLSxtrdefaultsubsequentplfmt
\end{verbatim}

In the event that any styles omit defining the newer \texttt{\GLSxtrfullformat} or \texttt{\GLSxtrfullplformat}, these are also initialised to defaults but ideally they should have their definitions provided.

The minimal set of commands that should have their definitions provided are the abbreviation plural suffix (\texttt{\abbrvpluralsuffix}) the display full forms: \texttt{\glsxtrfullformat}, \texttt{\glsxtrfullplformat} and their case-changing variants.

The inline full form commands only need to be provided if they behave differently from the display full form. The subsequent use commands only need to be provided if the default (only show the short form) isn’t suitable.
4. Abbreviations

The content of \textit{(display definitions)} is placed within the definition of an internal control sequence, so remember to use \textit{##} instead of \# to reference command parameters.

4.5.3.2.1. Suffix and Fonts

These are the generic suffix and font commands that vary according to the abbreviation style. The style should provide the appropriate definitions. The suffix should always be provided. The font commands are only required if the style applies any font formatting to either the long or short form.

\texttt{\textbackslash abbrvpluralsuffix initial: \textbackslash glsxtrabbrvpluralsuffix}

The plural suffix for the short form. For example, the \textit{long-short} style defines this to just use \texttt{\textbackslash glsxtrabbrvpluralsuffix}, but the smallcaps styles, such as \textit{long-short-sc} define this to \texttt{\textbackslash glsxtrscsuffix} in order to counteract the small caps font.

\texttt{\glsfirstabbrvfont\{⟨text⟩\}}

The font formatting command for the short form on first use. For example, the \textit{long-short-sc} style has:

\texttt{\renewcommand\*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}}

\texttt{\glsabbrvfont\{⟨text⟩\}}

The font formatting command for the short form. For example, the \textit{long-short-sc} style has:

\texttt{\renewcommand\*\glsabbrvfont[1]{\glsabbrvscfont{##1}}}

\texttt{\glsxtrrevert\{⟨text⟩\}}

This command is designed to counteract the effect of \texttt{\glsabbrvfont} if, for some reason, it shouldn’t be applied to part of the abbreviation. For example, you may prefer not to have
digits reduced with the smaller ("sm") styles.

\glsfirstlongfont{\text{⟨text⟩}}

The font formatting command for the long form on first use. For example, the \texttt{long-short-sc} style has:

\renewcommand*{\glsfirstlongfont}{\glsfirstlongdefaultfont{##1}}

\glslongfont{\text{⟨text⟩}}

The font formatting command for the long form. For example, the \texttt{long-short-sc} style has:

\renewcommand*{\glslongfont}{\glslongdefaultfont{##1}}

4.5.3.2.2. First Use Display Format

These commands always need to be provided.

\glsxtrfullformat{⟨entry-label⟩}{⟨insert⟩}

The singular display full form used on the first use of \texttt{\gls}.

\glsxtrfullplformat{⟨entry-label⟩}{⟨insert⟩}

The plural display full form used on the first use of \texttt{\glspl}.

\Glsxtrfullformat{⟨entry-label⟩}{⟨insert⟩}

The sentence case singular display full form used on the first use of \texttt{\Gls}.

\Glsxtrfullplformat{⟨entry-label⟩}{⟨insert⟩}
4. Abbreviations

The sentence case plural display full form used on the first use of \Glspl.

\GLSxtrfullformat{⟨entry-label⟩}{⟨insert⟩}

The all caps singular display full form used on the first use of \GLS.

\GLSxtrfullplformat{⟨entry-label⟩}{⟨insert⟩}

The all caps plural display full form used on the first use of \GLSp1.

4.5.3.2.3. Subsequent Use Display Format

These commands only need to be provided if the \gls-like commands don’t simply show the short form.

\glsxtrsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The singular form for subsequent use of \gls.

\glsxtrsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The plural form for subsequent use of \Glspl.

\Glsxtrsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The sentence case singular form for subsequent use of \Gls.

\Glsxtrsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The sentence case plural form for subsequent use of \GLSp1.

\GLSxtrsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The all caps singular form for subsequent use of \GLS.

\GLSxtrsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}
4. Abbreviations

The all caps plural form for subsequent use of \GLSpl.

The defaults all show the short form and insert encapsulated with the inner formatting \glsxtrgenentrytextfmt and \glsabbrvfont. The purpose of the inner formatting is to get it as close as possible to the actual text so \glsabbrvfont is placed outside of \glsxtrgenentrytextfmt.

The markshortwords attribute complicates matters as it inserts \glsxtrword and \glsxtrwordsep into the actual field value. In that case, the inner formatting is within \glsxtrword and \glsxtrwordsep, so only the insert material needs to be formatted.

If a custom style doesn’t need to support \innertextformat or \ifglsxtrinsertinside, it can reduce the complexity by omitting the inner formatting and conditionals, but this lack of support should be documented if the style is made generally available.

\glsxtrdefaultsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The default singular form for subsequent use of \gls.

\glsxtrdefaultsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The default plural form for subsequent use of \glsp1.

\Glssxtrdefaultsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The default sentence case singular form for subsequent use of \Glssxtr.

\Glssxtrdefaultsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The default sentence case plural form for subsequent use of \Glsp1.

\GLSxtrdefaultsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}

The default all caps singular form for subsequent use of \GLSxtr.

\GLSxtrdefaultsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}

The default all caps plural form for subsequent use of \GLSp1.
4. Abbreviations

4.5.3.2.4. Inline Full Format

These commands only need to be provided if the inline full form is different from the display full form.

\glsxtrinlinefullformat{(entry-label)}{(insert)}

The singular full form of \glsxtrfull.

\glsxtrinlinefullplformat{(entry-label)}{(insert)}

The plural full form of \glsxtrfullpl.

\Glsxtrinlinefullformat{(entry-label)}{(insert)}

The sentence case singular full form of \Glsxtrfull.

\Glsxtrinlinefullplformat{(entry-label)}{(insert)}

The sentence case plural full form of \Glsxtrfullpl.

\GLSxtrinlinefullformat{(entry-label)}{(insert)}

The all caps singular full form of \GLSxtrfull.

\GLSxtrinlinefullplformat{(entry-label)}{(insert)}

The all caps plural full form of \GLSxtrfullpl.

4.5.3.2.5. Wrapper Commands

These are commands that can be used in the definitions of the above to ensure that the appropriate accessibility fields and inner formatting is supported.

\glsxtrlongformat{(entry-label)}{(insert)}{(fmt-cs)}

This command is used in the definition of \glsxtrlong in some of the predefined abbreviation styles to format the long value of the entry identified by ⟨entry-label⟩ with the command ⟨fmt-cs⟩, which should take one argument.
4. Abbreviations

Accessibility support is implemented with \glsaccesslong if the markwords attribute is true otherwise with \glsaccessfmtlong using \glsxtrgenentrytextfmt for the inner formatting.

This is then encapsulated (including or excluding the \langle insert\rangle, according to \ifglsxtrinsertinside) with \langle fmt-cs\rangle. If the \langle insert\rangle content needs to be placed outside of \langle fmt-cs\rangle, it will be individually encapsulated with the inner formatting.

\Glsxtrlongformat{\langle entry-label\rangle}{\langle insert\rangle}{\langle fmt-cs\rangle}

As above, but sentence case.

\GLSxtrlongformat{\langle entry-label\rangle}{\langle insert\rangle}{\langle fmt-cs\rangle}

As above, but all caps.

\glsxtrlongplformat{\langle entry-label\rangle}{\langle insert\rangle}{\langle fmt-cs\rangle}

As \glsxtrlongformat, but for the longplural field.

\GLSxtrlongplformat{\langle entry-label\rangle}{\langle insert\rangle}{\langle fmt-cs\rangle}

As above, but sentence case.

\GLSxtrlongplformat{\langle entry-label\rangle}{\langle insert\rangle}{\langle fmt-cs\rangle}

As above, but all caps.

\glsxtrlongformatgrp{\langle entry-label\rangle}{\langle insert\rangle}{\langle fmt-cs\rangle}

As \glsxtrlongformat, but adds grouping around \langle insert\rangle (with the inner formatting inside the group).

\GLSxtrlongformatgrp{\langle entry-label\rangle}{\langle insert\rangle}{\langle fmt-cs\rangle}

176
As above, but sentence case.

\GLSxtrlongformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As above, but all caps.

\glsxtrlongplformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As \glsxtrlongplformat, but adds grouping around ⟨insert⟩ (with the inner formatting inside the group).

\Glsxtrlongplformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As above, but sentence case.

\GLSxtrlongplformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As above, but all caps.

\glsxtrshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

This command is used in the definition of \glsxtrshort and in some of the predefined abbreviation styles to format the \texttt{short} value of the entry identified by ⟨entry-label⟩ with the command ⟨fmt-cs⟩, which should take one argument.

Accessibility support is implemented with \glsaccessshort if the \texttt{markshortwords} attribute is true otherwise with \glsaccessfmtshort using \glsxtrgenentrytextfmt for the inner formatting.

This is then encapsulated (including or excluding the ⟨insert⟩, according to \ifglsxtrinsertinside) with ⟨fmt-cs⟩. If the ⟨insert⟩ content needs to be placed outside of ⟨fmt-cs⟩, it will be individually encapsulated with the inner formatting.

\Glsxtrshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As above, but sentence case.

\GLSxtrshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

177
As above, but all caps.

\glsxtrshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As \glsxtrshortformat, but for the \texttt{shortplural} field.

\GLSxtrshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As above, but sentence case.

\GLSxtrshortformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As above, but all caps.

\glsxtrshortformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As \glsxtrshortformat, but adds grouping around ⟨insert⟩ (with the inner formatting inside the group).

\GLSxtrshortformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As above, but sentence case.

\GLSxtrshortformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As above, but all caps.
4. Abbreviations

As above, but sentence case.

\GLSxtrshortplformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩}

As above, but all caps.

\glsxtrlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

A shortcut designed for ⟨long⟩ ⟨(short)⟩ styles. This is defined as:

\glsxtrlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}\glsxtrfullsep{⟨entry-label⟩}\glsxtrparen{\glsxtrshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨short-fmt-cs⟩}}

Note that the ⟨insert⟩ is only placed after the long form.

\GLGlsxtrlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but sentence case.

\GLSxtrlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but all caps.

\glsxtrlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As \glsxtrlongshortformat but uses the plural versions \glsxtrlongplformat and \glsxtrshortplformat.
4. Abbreviations

As above, but sentence case.

\GLSxtrlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but all caps.

\glsxtrshortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

A shortcut designed for ⟨short⟩ ⟨⟨long⟩⟩ styles. This is defined as:

\glsxtrshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨short-fmt-cs⟩}\glsxtrfullsep\gLsxtrparen{\gLsxtrlongformat{⟨entry-label⟩}{⟨long-fmt-cs⟩}}

Note that the ⟨insert⟩ is only placed after the short form.

The syntax is the same as for \gLsxtrlongshortformat even though \gLsxtrlongformat and \gLsxtrshortformat are flipped within the definition.

\Glsxtrshortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but sentence case.

\GLSxtrshortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but all caps.

\gLsxtrshortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As \gLsxtrshortlongformat but uses the plural versions \gLsxtrshortplformat and \gLsxtrlongplformat.
4. Abbreviations

As above, but sentence case.

\GLSxtrshortlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}

As above, but all caps.

4.6. Restoring Base Acronym Mechanism

It’s possible to revert \newacronym back to the definition provided by the base glossaries package. However, if you do this, you will lose all the abbreviation features provided by glossaries-extra.

Where possible, avoid this. If you’re simply trying to make the long form appear on first use with \newacronym, set the abbreviation style using:

\setabbreviationstyle[acronym]{long-short}

If you really need to use the original base glossaries package’s acronym mechanism, it’s better to stick with just glossaries and not use glossaries-extra. However, it may be that you need to use a glossaries-extra feature, such as \printunsrtglossary, but you have a custom acronym style that you can’t implement using the glossaries-extra abbreviation mechanism. This is a rare edge case for unusual formats, as it should be possible to implement most common abbreviation formats using the predefined styles.

Unpredictable results will occur if \RestoreAcronyms or \MakeAcronymsAbbreviations are used after abbreviations or acronyms have been defined.

\RestoreAcronyms

Restores \newacronym back to the original base glossaries interface. Note that this doesn’t affect \newabbreviation. It also sets the regular attribute for the acronym category to false and sets the acronym style to long-short (which is the default for the base package).

The display style for each glossary identified in the acronym lists is switched to the default
4. Abbreviations

acronym display style.

\MakeAcronymsAbbreviations

Counteracts \RestoreAcronyms.
5. Referencing (Using) Entries

Entries can be referenced using the \gls-like and \glstext-like commands, as described in the base glossaries manual. There are some additional commands provided by glossaries-extra:

- abbreviation commands, such as \glsxtrshort (see §4);
- commands for use in captions or section headings, such as \glstext (see §5.3.2);
- commands, such as \glsxtrp, designed for use within fields to help mitigate the problems of nesting (see §5.4);
- commands, such as \mglx, that are designed for referencing multi (or compound) entries (see §7);
- commands, such as \glsaccessname, used to incorporate accessibility support (see §9.2);
- commands, such as \dglx, that are designed for bib2gls’s dual entries (see §11.5.7);
- commands, such as \rglx, that depend on the number of entry records (see §11.4).

Additionally, the entry counting commands, such as \cgls, provided by the base glossaries package are modified by glossaries-extra (see §6.1).

The \gls-like commands are designed to produce text at that point in the document (the link text, §5.5), index the entry (to ensure that it appears in the glossary, §5.8) and unset the first use flag (which can alter the link text, §5.10). Additional information can be appended automatically with the post-link hook (§5.5.4). The link text is given by the entry style (see §5.5.5) or by the final argument of \glsdisp.

The \glstext-like commands are designed to produce text at that point in the document (the link text, §5.5) and index the entry (to ensure that it appears in the glossary, §5.8). Additional information can be appended automatically with the post-link hook (§5.5.4). The link text is determined by the calling command. For example, the corresponding field value (possibly encapsulated with \glstextregularfont and the inner formatting) for commands like \glstext or the final argument of \glslink.

The \gls-like and \glstext-like commands can all be used with a star (*) or plus (+) modifier. The star modifier automatically implements hyper=false (disables the hyperlink) and the plus modifier automatically implements hyper=true (forces the hyperlink on, if supported).
5. Referencing (Using) Entries

With glossaries-extra, it’s possible to define an additional modifier for your own use with:

\GlsXtrSetAltModifier{⟨token⟩}{⟨options⟩}

The ⟨token⟩ must be a single token, so a multi-byte UTF-8 character will require a native Unicode engine (XƎLÂTEX or LuaLÂTEX). For example, the following:

\GlsXtrSetAltModifier{!}{format=glsignore}

means that \gls{⟨label⟩} will be equivalent to \gls[format=glsignore]{⟨label⟩}.

It’s also possible to redefine the star and plus modifiers:

\GlsXtrSetStarModifier{⟨options⟩}

This sets the options to use for the star modifier.

\GlsXtrSetPlusModifier{⟨options⟩}

This sets the options to use for the plus modifier. For example, the following:

\GlsXtrSetPlusModifier{noindex}

means that the star modifier will now suppress indexing instead of switching on the hyperlink.

The \gls-like and \glstext-like commands have a complicated internal structure, which can be viewed as a series of layers. The outermost common layer is:

\% save settings
\% initialise options, see §5.1
\glslinkwrcontent{⟨index & fmt content⟩}
\% restore settings
\% post-link hook, see §5.5.4

The ⟨index & fmt content⟩ consists of the indexing (see §5.8) and the (possibly hyperlinked)
Referencing (Using) Entries

formatted text, see §5.5. The \( \langle \text{index} \& \text{fmt content} \rangle \) code is encapsulated with:

\[
\text\{\text\color{red}{\text\glslinkwrc}\langle\text{code}\rangle}\]

In v1.48, this was added to scope the link text and indexing code, which helped to prevent unwanted spacing caused by the whatsit and also helped to prevent some setting leakage, in the event of nesting (see §5.4), but this caused spacing issues when used in math mode, so from v1.49 this command now simply does its argument. The whatsit is now scoped with \glsencapwrc instead.

The \glsxtrp command, designed for nested use, deals with the problem by suppressing the post-link hook and adding an outer group. For example, \glsxtrp{short}{html} behaves like:

\[
\{\let\glspostlinkhook\relax \glsxtrp[\text{noindex, nothing}]{html}\}
\]

Note that the code to suppress the post-link hook has been moved to \glsxtrpInit, so it is now possible to allow the post-link hook but it won’t be able to lookahead beyond the added outer group.

Depending on the settings (the \texttt{wrgloss} option or the \texttt{wrgloss} attribute), the indexing may come before the text:

\[
\text\{\text\color{red}{\text\glslinkwrc}\langle\text{index}\rangle\langle\text{fmt content}\rangle}\]

or after the text:

\[
\text\{\text\color{red}{\text\glslinkwrc}\langle\text{fmt content}\rangle\langle\text{index}\rangle}\]

or may be suppressed with \texttt{noindex}:

\[
\text\{\text\color{red}{\text\glslinkwrc}\langle\text{fmt content}\rangle}\]

The \( \langle \text{fmt content} \rangle \) part is described in §5.5. The \( \langle \text{index} \rangle \) part is the actual indexing (see §5.8) but also increments the index count, if applicable. Both the associated whatsit and increment are encapsulated with \glsencapwrc.

Avoid using \glsstxt, \glspl, \glsfi and \glsfipl (and their case-changing variants) with entries that have been defined with \texttt{newabbreviation}. Some of the abbreviation styles are too complicated to work with those commands. Instead, use commands like \glsxtrp, \glsxtrf or use \texttt{gls} with the \texttt{prereset} or \texttt{preunset} options.
5. Referencing (Using) Entries

The base glossaries package provides a way to adjust the formatting of the link text for the \gls-like commands according to the glossary type with \defglsentryfmt. The glossaries-extra package changes the default entry formatting (§5.5.5) and provides additional ways of modifying the displayed content (§5.5).

The heading commands (described in §5.3) are designed to prevent indexing or changes to the first use flag if they appear in the table of contents (or list of figures, etc) or if they appear in the page header.

Although the base glossaries package warns against nested link text, the glossaries-extra package provides \glsxtrp which can be used instead of \gls in field values to overcome some of the associated problems. See §5.4 for further details.

If you need to simply access a field value without any formatting, see §5.11. (See §3.5 to set field values.) If you want to encapsulate the value with the appropriate accessibility tag, see §9.2.

Commands such as \glsadd (see §5.8) and \glsssee (see §5.9) are designed to only index (to ensure the entry appears in the glossary) without producing any text or changing the first use flag.

The \gls-like, \glstext-like and \glsadd commands all have an initial optional argument that can be used to override the default actions. Some options are only applicable for particular subsets of referencing commands. For example, noindex is pointless for \glsadd since the sole purpose of that command is to index. Whereas types is only available with \glsaddall.

5.1. Options

Some options are provided by the base glossaries package, but there are some additional options provided by glossaries-extra, which are listed in §5.1.2. Below, \langle option-list \rangle indicates the options that are passed in the optional argument of the calling command (such as \gls).

The order that the options are applied is:

1. prereset, preunset and postunset options are initialised by \glsinitreunsets;
2. hyper is initialised by \glsxtrchecknohyperfirst (\glsfirst-like only);
3. wrgloss option is initialised by \glsxtrinitwrgloss (not implemented by \glsadd or \glsxtrfmt);
4. hyperoutside option is initialised by \glsxtrinithyperoutside (not implemented by \glsadd or \glsxtrfmt);
5. initialise noindex=false (not \glsadd);
6. options identified by \GlsXtrSetDefaultGlsOpts, \GlsXtrAppToDefaultGlsOpts or \GlsXtrPreToDefaultGlsOpts (not implemented by \glsadd);
7. (\glsxtrfmt only) options provided in \GlsXtrFmtDefaultOptions;
5. Referencing (Using) Entries

8. (\gls-like only) the hyperfirst package option, nohyperfirst attribute and nohypernext attributes are checked to determine if the hyper option should be switched off (tests followed by \glslinkcheckfirsthyperhook);

9. \glslinkpresetkeys (not implemented by \glsadd or \glsxtrfmt);

10. (\glsadd only) \glsaddpresetkeys;

11. (option-list);

12. \glslinkpostsetkeys (provided by the base glossaries package, not implemented by \glsadd or \glsxtrfmt);

13. (\glsadd only) \glsaddpostsetkeys.

5.1.1. Setting Up Defaults

You can (locally) set your preferred default options for the \gls-like and \glstext-like commands using:

\GlsXtrSetDefaultGlsOpts{⟨options⟩}

The ⟨options⟩ may be any options that you can pass to those commands. These options also apply to \glsxtrfmt but not to \glsadd.

Note that multiple instances of \GlsXtrSetDefaultGlsOpts will override each other.

If you want to add to the existing options, you can use one of the following commands (both may be scoped).

\GlsXtrAppToDefaultGlsOpts{⟨options⟩}

Appends ⟨options⟩ to the list of default options.

\GlsXtrPreToDefaultGlsOpts{⟨options⟩}

Prepends ⟨options⟩ to the list of default options.

For example, to prevent indexing in the front matter and back matter but not in the main matter:
5. Referencing (Using) Entries

\frontmatter
\GlsXtrSetDefaultGlsOpts{noindex}
...
\mainmatter
\GlsXtrSetDefaultGlsOpts{ }
...
\backmatter
\GlsXtrSetDefaultGlsOpts{noindex}

Note that \texttt{noindex=false} is now set before the options given in \texttt{\GlsXtrSetDefaultGlsOpts} to ensure that the setting is correctly initialised, so as from v1.49 you can simply set an empty options list to reset the default. Prior to v1.49, it was necessary to ensure that the \texttt{noindex} key was always present in the options list to avoid instability. So for pre v1.49, the line after \texttt{\mainmatter} in the above would need to be:

\GlsXtrSetDefaultGlsOpts{noindex=false}

The default location \texttt{encap} is \texttt{glsnumberformat} but can be changed (locally) with:

\GlsXtrSetDefaultNumberFormat{\langle encap \rangle}

This can be overridden by explicitly setting the \texttt{format} key.

The default options for \texttt{\glsxtrfmt} only are given by:

\GlsXtrFmtDefaultOptions \textit{initial: noindex}

This command should simply expand to the required list of options. These options are set after any options given in \texttt{\GlsXtrSetDefaultGlsOpts} and before \texttt{\langle option-list \rangle}.

\glslinkpresetkeys

This hook is performed after any settings provided in \texttt{\GlsXtrSetDefaultGlsOpts} but before \texttt{\langle option-list \rangle}. This hook also applies to \texttt{\glsxtrfmt} but not to \texttt{\glsadd}.

Note that \texttt{\glslinkpostsetkeys}, provided by the base glossaries package, is performed after \texttt{\langle option-list \rangle} is processed.

\glsaddpresetkeys
5. Referencing (Using) Entries

This hook, which is only used by \glsadd, is performed before \textit{\{option-list\}}.

\texttt{\glsaddpostsetkeys}

This hook, which is only used by \glsadd, is performed after \textit{\{option-list\}}.

\texttt{\glsinitreunsets}

This hook initialises the pre unset/reset options to: \texttt{prerest=none} and \texttt{preunset=none}. It also initialises the \texttt{postunset} setting to perform the post-unset (where applicable) but it will retain the current local/global setting.

This hook will also implement the local repeat unset feature of \texttt{\GlsXtrUnsetBuffer\-EnableRepeatLocal}.

\texttt{\glsxtrchecknohyperfirst{\langle entry-label \rangle}}

This hook is only used by \glsfirst, \glsfirstplural and their case-changing variants. The hook will implement \texttt{hyper=false} if the \texttt{nohyperfirst} attribute is set to \texttt{true}.

\texttt{\glsxtrinitwrgloss}

This hook initialises the default setting of the \texttt{wrgloss} option. If the \texttt{wrgloss} attribute is set to \texttt{after} then this implements \texttt{wrgloss=after} otherwise it implements \texttt{wrgloss=before}. This setting can subsequently be overridden by \texttt{\GlsXtrSetDefaultGlsOpts}, \texttt{\glslink-presetkeys}, the \texttt{\{option-list\}} argument or \texttt{\glslinkpostsetkeys}. This hook also applies to \texttt{\glsxtrfmt} but not to \texttt{\glsadd}.

If you prefer to have the default to place the indexing after the link text, you can redefine this hook as follows:

\texttt{\renewcommand*{\glsxtrinitwrgloss}{\%}
\glsifattribute{\glslabel}{wrgloss}{before}{%}
% \glsxtrinitwrglossbeforetrue
%
% \glsxtrinitwrglossbeforefalse
%
}}

189
5. Referencing (Using) Entries

\glsxtrinithyperoutside

This hook initialises the default setting of the `hyperoutside` option. If the `hyperoutside` attribute is set to `false` then this implements `hyperoutside=false` otherwise it implements `hyperoutside=true`. This setting can subsequently be overridden by \GlsXtrSetDefaultGlsOpts, \glslinkpresetkeys, the \{option-list\} argument or \glslinkpostsetkeys. This hook also applies to \glsxtrfmt but not to \glsadd.

Within any of the hooks that are used by the \gls-like, \glstext-like or \glsxtrfmt commands, you can set options using:

\setupglslink\{\(\text{options}\)\}

Within any of the hooks that are used by \glsadd, you can set options with:

\setupglsadd\{\(\text{options}\)\}

5.1.2. Additional Options

Options for the \gls-like and \glstext-like commands that are provided by the base glossaries package also apply to new commands like \glsxtrfmt and \glsfmttext. In addition, the options below are provided by glossaries-extra. Note that some options, such as `postunset`, only apply to the \gls-like commands. Options that relate to the hyperlink, formatting, first use flag or whether/where (noindex/wrgloss) to perform indexing aren’t available for \glsadd.

`hyperoutside=\langle boolean\rangle`  \hspace{1cm} \textit{default: true; initial: true}

This boolean option determines whether the hyperlink should be inside or outside of \glstextformat (see §5.5.1). If true, the link text is encapsulated as:

\langle hyperlink-cs\rangle\{\langle target\rangle\}\{\glstextformat\{\langle text\rangle\}\}

otherwise it’s encapsulated as:

\glstextformat\{\langle hyperlink-cs\rangle\{\langle target\rangle\}\{\langle text\rangle\}\}
where \textit{\texttt{hyperlink-cs}} is the command that generates the hyperlink (if enabled).

\begin{verbatim}
\textbf{textformat}=\langle\texttt{csname}\rangle
\end{verbatim}

The value of this key should be the name of a control sequence (without the leading backslash). If this option is set, the given control sequence will be used instead of \texttt{\textbackslash glstextformat} to encapsulate the link text. Note that this control sequence should take a single argument (the link text). See §5.5.1 for further details.

This option will override the \texttt{textformat} attribute.

\begin{verbatim}
\textbf{innertextformat}=\langle\texttt{csname}\rangle initial: \texttt{glsxtrdefaultentrytextfmt}
\end{verbatim}

The value of this key should be the name of a control sequence (without the leading backslash). The command \texttt{\textbackslash glsxtrgenentrytextfmt} (which shouldn’t be redefined) is assigned to this control sequence at the start of the \texttt{\textbackslash gls}-like and \texttt{\textbackslash glstext}-like commands. This command is used within the predefined abbreviation styles and within \texttt{\textbackslash glsgenentryfmt} to encapsulate the entry field values.

Custom styles that don’t use \texttt{\textbackslash glsxtrgenentrytextfmt} won’t support this key. See §5.5.3 for further details.

Some formatting commands require direct access to the actual text or else the content has to be placed inside a box (which inhibits line-breaking). These commands won’t work with \texttt{\textbackslash textformat} as the text is usually too deeply embedded. This option provides a way of using those problematic commands, however there’s still no guarantee that they will work (for example, in the case of custom styles or where the field value itself contains commands).

\begin{verbatim}
\textbf{postunset}=\langle\textbf{value}\rangle default: \texttt{global}; initial: \texttt{global}
\end{verbatim}

This option only applies to the \texttt{\textbackslash gls}-like commands and indicates whether or not to unset the first use flag after the link text. It may take one of three values: \texttt{global} (behaves like \texttt{local=false}), \texttt{local} (behaves like \texttt{local=true}) or \texttt{none} (doesn’t unset the first use flag after the first use). See §5.10.

\begin{verbatim}
\textbf{prerreset}=\langle\textbf{value}\rangle default: \texttt{local}; initial: \texttt{none}
\end{verbatim}
5. Referencing (Using) Entries

This option may take one of three values: none (no reset), local or global. This option (if not none) will reset the first use flag before the link text and additionally change \glsxtrifwasfirstuse so that it indicates that this was the first use of the entry. See §5.10.

Note that this is different from using \glslocalreset or \glsreset before the \glstext-like commands. Normally \glstext and \glsplural will define \glsxtrifwasfirstuse so that it indicates that this was not the first use of the entry (regardless of whether or not the entry has actually been used).

For example:

\glsdefpostlink{general}{\glsxtrpostlinkAddDescOnFirstUse}
\newglossaryentry{sample}{name={sample},
  first={sample first use},description={an example}}
\begin{document}
Text field: \glstext{sample}.

First use: \gls{sample}. Next use: \gls{sample}.

Force reset: \gls[prereset]{sample}.
Used? \ifglsused{sample}{Yes}{No}.

Force reset: \glstext[prereset]{sample}.
Used? \ifglsused{sample}{Yes}{No}.
\end{document}

Example 90: Illustrating the prereset option

Text field: sample.
First use: sample first use (an example). Next use: sample.
Force reset: sample first use (an example). Used? Yes.

Note that \gls unsets the first use flag (unless postunset=none), so the sample entry is marked as used afterwards, but \glstext doesn’t alter the first use flag, after the link text so the sample entry is still marked as unused afterwards.

\texttt{\textbackslash preunset=⟨value⟩}

\textit{default: local; initial: none}

This option may take one of three values: none (no unset), local or global. This option (if not none) will unset the first use flag before the link text and additionally change \glsxtrifwasfirstuse so that it indicates that this wasn’t the first use of the entry. See §5.10.
The \texttt{preunset} key is always performed after the \texttt{prereset} key.

Note the effect of using a global reset but a local unset in the example below. Both options are performed, but the unset locally overrides the global reset.

\begin{verbatim}
\glsdefpostlink{general}{\glsxtrpostlinkAddDescOnFirstUse}
\newglossaryentry{sample}{name={sample},
  first={sample first use},description={an example}}
\begin{document}
\gls{sample}. Used? \ifglsused{sample}{Yes}{No}.

{\glsfirst[preunset=local,prereset=global]{sample}. Used? \ifglsused{sample}{Yes}{No}.}

Used? \ifglsused{sample}{Yes}{No}.

{\gls[preunset=local,prereset=global]{sample}. Used? \ifglsused{sample}{Yes}{No}.}

Used? \ifglsused{sample}{Yes}{No}.
\end{document}
\end{verbatim}

Example 91: Combining \texttt{prereset} and \texttt{preunset}

\begin{verbatim}
\gls{sample first use (an example)}. Used? Yes.
\end{verbatim}

Remember that \gls{} globally unsets the first use flag (unless changed with \texttt{postunset}), which counteracts \texttt{prereset=global}.

\textbf{noindex} = \texttt{\{boolean\}} \hspace{1cm} \textit{default: true; initial: false}

This is a boolean option that determines whether or not to suppress the normal indexing. For example, to prevent any locations in the front matter or back matter appearing in the glossary:
Note that if you are using auto-indexing (see §12), `noindex=false` will also suppress the auto-indexing.

If you are using `bib2gls`, you may want to consider instead using `format=glsignore` to create an ignored location that ensures the entry is selected without adding a location to the location list. (Don’t use this method for the other indexing methods as you’ll end up with invisible locations with spurious commas in your location lists.)

This option may take one of two values, before or after, which indicate whether the indexing should occur before or after the link text. The indexing creates a whatsit that can interfere with spacing or cause other problems. The other thing to consider is where the link text is long, such as a phrase or full form of an abbreviation, that may be split by a page break. You will need to decide if you want the indexing before the link text, so that the location is at the end of the page where the text starts, or if you want the indexing after the link text, so that the location is at the start of the next page where the text ends.

This option corresponds to a conditional:

The hook \glsxtrinitwrgloss sets this conditional according to whether or not the wrgloss attribute has been set to after:
5. Referencing (Using) Entries

\texttt{thevalue}=(\texttt{location})

Sets the entry location to the given value instead of obtaining it from the location counter. If you are using \texttt{hyperref} you may also need to set the location’s hypertarget with \texttt{theHvalue}.

This option is primarily intended for use with \texttt{bib2gls} to supply locations that don’t have an associated counter within the document, such as an external location. If you want to automatically add locations from a supplemental document to an entry’s location list, you can use the \texttt{supplemental-locations} resource option. See the \texttt{bib2gls} user manual for further details.

For example, to index a location in a supplementary document:

\begin{verbatim}
\glsadd[thevalue={Suppl. 2.45}]{sample}
\end{verbatim}

This will add “Suppl. 2.45” to the location list for the “sample” entry.

Note that the value must conform to the indexing application’s location syntax. For \texttt{makeindex}, this is limited to \texttt{Roman}, \texttt{roman}, \texttt{arabic}, \texttt{alph} and \texttt{Alph}. With \texttt{xindy}, the location syntax must be defined in the \texttt{xindy} module (standard location syntaxes are supplied by default). There’s no restriction on the location syntax for \texttt{bib2gls}, although if it can’t deduce a numerical value it won’t be able to form a range.

If you want a hyperlink to an external file, you can use:

\begin{verbatim}
\glsxtrsupphypernumber{⟨location⟩}
\end{verbatim}

as the formatting command for the location encap. For example:

\begin{verbatim}
\glsadd[thevalue=S.2,format=glsxtrsupphypernumber]{sample}
\end{verbatim}

The path to the external file needs to be set in the \texttt{externallocation} category attribute.

The hyperlink for the supplementary location may or may not take you to the relevant place in the external PDF file \textit{depending on your PDF viewer}. Some may not support external links, and some may take you to the first page or last visited page.
For example, if both `sample-suppl-hyp.pdf` and `sample-suppl-main-hyp.pdf` are in the same directory, then viewing `sample-suppl-main-hyp.pdf` in Evince will take you to the correct location in the linked document (when you click on the S.2 external link), but Okular will take you to the top of the first page of the linked document.

This method can only be used where there is one external source for the designated category (identified by the `externallocation` attribute). For multiple sources, you need to use `bib2gls v1.7+`, which is the better method in general as it can automatically fetch the relevant locations from the aux files of the designated external documents without the need to explicitly use `\glsadd`.

```
\theHvalue=⟨the-H-value⟩
```

Sets the hypertarget corresponding to the location, which will be used if the `format` supports hyperlinks. This is analogous to `hyperref`'s `\the⟨counter-name⟩` that provides the hypertarget for a reference to `\the⟨counter-name⟩`.

This option is primarily intended for use with the `thevalue` option.

Unless you are using `record=nameref`, you must ensure that it’s possible to form `⟨the-H-value⟩` from `⟨h-prefix⟩⟨thevalue⟩` for some `⟨h-prefix⟩` (where `⟨thevalue⟩` is given by `thevalue` or the value of the location counter). This restriction is due to the limitations imposed by `makeindex` and `xindy`.

```
prefix=⟨link-prefix⟩
```

This option locally redefines `\gloolinkprefix` to `⟨link-prefix⟩`. If you are using `\printunsrtglossary` to redisplay a list (possibly in a different order) then you will need some way of changing the entry targets to avoid duplicate hyperlink targets. One way of achieving this is to redefine `\gloolinkprefix` for the subsequent lists. You will then need to use the `prefix` option in commands like `\gls` to ensure that the hyperlink for the link text points to the desired list.

This option is intended for use with the “unsrt” family of commands and `\glsxtrcopytogonalssary` (which is used by `bib2gls`). The other indexing methods don’t support repeated lists.
5.2. Case Changing

Case-changing commands, such as \Gls and \GLS, perform the conversion using commands provided by mfirstuc. The underlying commands provided by mfirstuc were redesigned in v2.08 to use the newer, better case-changing commands available with the \LaTeX3 kernel. The base glossaries package v4.50 and glossaries-extra v1.49 were developed concurrently with mfirstuc v2.08 to take advantage of the new features. Version 1.49 of glossaries-extra was also developed concurrently with bib2gls v3.0 which, in turn, was developed alongside version 0.9.2.7b of the \TeX{} parser library.

It’s not possible to upload all these new versions at the same time, so it will be necessary to stagger their deployment. The new case-changing features will work best when all these new versions are installed. In the interim, a reduced feature set will be used.

5.2.1. Sentence Case Commands

Both the base glossaries package and the glossaries-extra package provide sentence case commands, which convert the first letter to uppercase. These are provided for situations where an entry is referenced at the start of a sentence. Sentence-casing is also implemented when the attributes \texttt{glossname} or \texttt{glossdesc} are set to \texttt{firstuc}.

The case conversion is performed using:

\texttt{\textbackslash glssentencecase\{$\langle \text{text} \rangle$\}}

The default definition uses \texttt{\textbackslash makefirstuc}, which is provided by the \texttt{mfirstuc} package. This was originally part of the base glossaries package, but was split into a separately distributed package in 2015. Back then, there was no expandable sentence-case command. There was also a problem with referencing entries where link text was encapsulated with a text-block command (which occurs, in particular, with acronym and abbreviation styles). The first letter of the text-block command’s argument needed to be obtained, which resulted in some trickery that proved problematic with UTF-8.

The \LaTeX3 kernel now provides a suitable expandable command that works with UTF-8, and mfirstuc v2.08+ provides \texttt{\textbackslash MFUsentencecase} that directly interfaces with it. If an older version of mfirstuc is installed, glossaries v4.50+ and glossaries-extra v1.49+ will provide \texttt{\textbackslash MFUsentencecase}. You can use this in expandable contents. For example:

\texttt{\section{\textbackslash MFUsentencecase\{$\text{\textbackslash glsentrytext\{label\}}$\}}}

However, in the above example, it’s simpler to do:

\texttt{\section{\textbackslash Glsfmttext\{label\}}}
5. Referencing (Using) Entries

If hyperref has been loaded, \Glsfmttext{label} will now expand to:

\MFUsentencecase{\glsentrytext{label}}

in the PDF bookmark.

Internally, \makefirstuc now uses \MFUsentencecase to perform the case conversion, but it still parses its argument to determine if it starts with \langle cs}{(text)\rangle. This means that with mfirstuc v2.08+, you now don’t have to worry about UTF-8 characters occurring at the start of the text.

For example, with mfirstuc v2.07 you would need to do something like:

\newglossaryentry{elite}{name={{é}lite},
\description={...}}

in order for \Gls{elite} to work. Whereas with mfirstuc v2.08, you can now simply do:

\newglossaryentry{elite}{name={élite},
\description={...}}

(As from glossaries v4.47, it should be possible to use UTF-8 characters in the label as well.)

Whilst you can redefine \glssentencecase to use \MFUsentencecase directly (without using \makefirstuc as an intermediary), this may result in content being expanded that wouldn’t have been expanded previously. In particularly, if \langle cs\rangle isn’t robust and expands to content that includes labels then the case-change can fail. You also won’t be able to take advantage of the blockers and mappings that are only recognised as such by \makefirstuc.

If you use \MFUsentencecase instead, blockers and mappings will be treated as exclusions, which are likely to result in unwanted side-effects.

Both \makefirstuc and \MFUsentencecase recognise exclusions. These are text-block commands which take a single mandatory argument that needs to be skipped. For example, in the following \glsadd{example} needs to be skipped:

\MFUsentencecase{\glsadd{example}some text}

Exclusions are identified with \MFUexcl. If you have an older version of mfirstuc, this won’t be defined, so glossaries v4.50+ and glossaries-extra v1.49+ provide:

\glsmfuexcl{(cs)}

This will use \MFUexcl with mfirstuc v2.08+. With older versions, a definition will be provided that works with \MFUsentencecase, but exclusions won’t be recognised by \make-
5. Referencing (Using) Entries

firstuc.

As from glossaries v4.50, \glsadd will be identified as an exclusion (via \glsmfuexcl), but the optional argument will cause a problem if present. See the mfirstuc v2.08+ manual for a workaround. Note that commands such as \glsaddall and \glsaddeach aren’t identified as exclusions as they aren’t expected to occur in text that may require a case-change.

With glossary entry references, there are commands that take a label as the argument, which shouldn’t have any case-changed applied, but also shouldn’t be skipped. For example:

\begin{verbatim}
\makefirstuc{\GLS{example} something}
\end{verbatim}

In this situation, there shouldn’t be any case-change as \GLS already implements a case-change. This type of command is referred to as a blocker in the mfirstuc manual, as it indicates a command that should prevent any case-change if it’s encountered at the start of the text. Blockers are identified with \MFUblocker. If you have an older version of mfirstuc, this won’t be defined, so glossaries v4.50+ and glossaries-extra v1.49+ provide:

\begin{verbatim}
\glsmfublocker{⟨cs⟩}
\end{verbatim}

This will use \MFUblocker with mfirstuc v2.08+. With older versions, it will simply use \gls-mfuexcl which will instead identify the command as an exclusion and won’t be recognised by \makefirstuc. See the mfirstuc v2.08+ manual for further information about blockers.

As from glossaries v4.50+, commands like \GLS will be identified as blockers using \glsmfublocker, and glossaries-extra now identifies similar commands, such as \rGLS as blockers.

Finally, there are mappings. These are commands that should be substituted with another command, which is expected to perform the case-change. For example:

\begin{verbatim}
\makefirstuc{\gls{example} something}
\end{verbatim}

This shouldn’t skip or block \gls but instead should convert the text to:

\begin{verbatim}
\Gls{example} something
\end{verbatim}

This is implemented by adding a mapping from \gls to \Gls. Mappings are added using \MFUaddmap. If you have an older version of mfirstuc, this won’t be defined, so glossaries v4.50+ and glossaries-extra v1.49+ provide:

\begin{verbatim}
\glsmfuaddmap{⟨cs1⟩}{⟨cs2⟩}
\end{verbatim}
This will use `\MFUaddmap` with `mfirstuc v2.08+`. With older versions, it will simply use `\gls-mfuexcl` which will instead identify the command as an exclusion and won’t be recognised by `\makefirstuc`. See the `mfirstuc v2.08+` manual for further information about mappings.

As from `glossaries v4.50+`, commands like `\gls` will be mapped to the appropriate sentence case command using `\glsMFUaddmap`, and `glossaries-extra` now identifies similar mappings, such as `\rgls` mapped to `\rGls`.

In order to integrate the full set of features provided by `mfirstuc v2.08+`, you will need both `glossaries v4.50+` and `glossaries-extra v1.49+`.

### 5.2.2. Lower Case

\[
\texttt{\textbackslash glslowercase\{⟨text⟩\}}
\]

This is defined by `glossaries v4.50+` to use the \LaTeX3 command to convert to lowercase. If an older version of `glossaries` is present, then this command will be provided by `glossaries-extra` but it will be defined to use `\MakeTextLowercase` instead. This command is primarily provided for use with small caps styles to convert an abbreviation to lowercase, but isn’t actually used anywhere by default.

### 5.2.3. Upper Case

\[
\texttt{\textbackslash glsupercase\{⟨text⟩\}}
\]

This is defined by `glossaries v4.50+` to use the \LaTeX3 command to convert to uppercase (all caps). If an older version of `glossaries` is present, then this command will be provided by `glossaries-extra` but it will be defined to just use `\mfirstucMakeUppercase`, which is provided by `mfirstuc`. This command is used by all caps commands such as `\GLSxtrusefield`.

### 5.2.4. Title Case

\[
\texttt{\textbackslash glscapitalisewords\{(content)\}}
\]

This is defined by `glossaries v4.48` to use `\capitalisewords` to convert to title case. If you experience any errors with title case commands, such as `\glsentrytitlecase`, or attributes such as `glossdesc` then try redefining this command to use `\capitalisefmtwords*` instead. See the `mfirstuc` manual for further details.
5.3. Entries in Sectioning Titles, Headers, Captions and Contents

The glossaries user manual cautions against using commands like `\gls` in chapter or section titles. The principle problems are:

- if you have a table of contents, the first use flag will be unset in the contents rather than later in the document;
- if you have the location lists displayed in the glossary, unwanted locations will be added to it corresponding to the table of contents (if present) and every page that contains the entry in the page header (if the page style in use adds the chapter or section title to the header);
- if the page style in use adds the chapter or section title to the header and attempts to convert it to uppercase, the entry label (in the argument of `\gls` etc) will be converted to uppercase and the entry won’t be recognised;
- if you use `hyperref`, commands like `\gls` can’t be expanded to a simple string and only the label will appear in the PDF bookmark (with a warning from `hyperref`);
- if you use `hyperref`, you will end up with nested hyperlinks in the table of contents.

Similar problems can also occur with captions (except for the page header and bookmark issues).

The `glossaries-extra` package tries to resolve the header problem by modifying `\markright`, `\markboth` and `\@starttoc`. If this causes unwanted side-effects, you can restore their former definitions using:

```
\glsxtrRevertMarks
```

This will revert `\markright`, `\markboth` and `\@starttoc` back to the definitions in effect when `glossaries-extra` was loaded. Alternatively, you can use:

```
\glsxtrRevertTocMarks
```

This will only revert `\@starttoc`.

If you use `\glsxtrRevertMarks` or `\glsxtrRevertTocMarks`, you will need to employ the simplistic approach, described in §5.3.1, which is the method recommended by the glossaries user manual. Otherwise, you can use the commands described in §5.3.2, which provide a better solution.
5. Referencing (Using) Entries

5.3.1. Simplistic Approach

To get around all these problems, the glossaries user manual recommends using the expandable non-hyperlink commands, such as \glsentrytext (for regular entries) or \glsentryshort (for abbreviations). This is the simplest solution, but doesn’t allow for special formatting that’s applied to the entry through commands like \glstext or \glsxtrshort.

For example:

```latex
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\glsdefpostlink{general}{\glsxtrpostlinkAddDescOnFirstUse}
\newglossaryentry{sample}{name={sample},
 description={an example}}
\newglossaryentry{alpha}{name={\ensuremath{\alpha}},description={alpha}}
\begin{document}
\tableofcontents
\section{\texorpdfstring{\Glsentrytext{sample} and \glsentrytext{alpha}}{Sample and alpha}}
First use: \gls{sample} and \gls{alpha}.
Next use: \gls{sample} and \gls{alpha}.
\printunsrtglossary
\end{document}
```
5. Referencing (Using) Entries

Example 92: References in section headings (simplistic approach)

Contents

1 Sample and $\alpha$ 1

Glossary 1

1 Sample and $\alpha$

First use: sample (an example) and $\alpha$ (alpha).
Next use: sample and $\alpha$.

Glossary

sample an example
$\alpha$ alpha

This solves some problems: it avoids nested links in the table of contents, the first use flag isn’t prematurely unset and the PDF bookmarks has a reasonable substitution, but it still isn’t a complete solution as the above document will fail if the page style is changed to headings and a page break is inserted before the section (after `\tableofcontents`), which will lead to the error:

Glossary entry `SAMPLE` has not been defined.

This is because the case-change applied to the header converts the label “sample” to “SAMPLE”, which doesn’t correspond to a defined entry. (This can now be avoided with `mfirstuc v2.08+`.)

If the case conversion is applied by, then the case-change can be prevented by encapsulating the label with `\NoCaseChange`, but this ends up quite complicated. This is actually what the commands describe in §5.3.2 do when they are in a heading. This allows for older versions of `mfirstuc` that don’t recognise exclusions. See §5.2 for further details.

The `\NoCaseChange` command was originally provided by the `textcase` package to prevent `\MakeTextUppercase` from applying a case-change. The functionality of the `textcase` package has now been absorbed into the `\LaTeX` kernel, which means that as from 2022, `textcase` is deprecated and `\NoCaseChange` is defined by the kernel.
5. Referencing (Using) Entries

5.3.2. New Commands Designed for Chapter/Section Headings or Captions

This section is irrelevant if you use \glsxtrRevertMarks to restore the definitions of \markright, \markboth and \@starttoc. If you use \glsxtrRevertTocMarks, then this section is only applicable to \markright and \markboth.

The commands listed here are provided for use within captions or section headings. They are designed to overcome some of the problems illustrated in the previous section. Note that they only have a single argument, the entry label. There are no optional arguments. Below, “header” refers to page header text added with \markright or \markboth, and “contents” refers to the table of contents or any other “list of” that uses \@starttoc, such as the list of figures.

Each command \glsfmt⟨field⟩ (such as \glsfmttext or \glsfmtshort) behaves like an analogous \gls⟨field⟩ or \glsxtr⟨field⟩ command (such as \glstext or \glsxtrshort) but with the options noindex and hyper=false and no insert. When they occur within a header, they are protected from having any case-change applied (which will interfere with the entry label). Since this means they won’t appear in all caps in the header, the head-uc attribute may be set to use the all caps \GLS⟨field⟩ or \GLSxtr⟨field⟩ instead (such as \GLStext or \GLSxtrshort).

There is currently only support for the name, text, plural, first, firstplural, short, shortplural, long, and longplural fields, and also limited support for the full form of abbreviations. For other fields, you will need to follow the recommendation of the glossaries manual (as discussed above in §5.3.1).

The previous example can be rewritten as follows:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\glsdefpostlink{general}{\glsxtrpostlinkAddDescOnFirstUse}
\newglossaryentry{sample}{name={sample},description={an example}}
\newglossaryentry{alpha}{name={\ensuremath{\alpha}},description={\ensuremath{\alpha}}} 
\begin{document}
\tableofcontents
\section{\glsfmttext{sample} and \glsfmttext{alpha}}

First use: \gls{sample} and \gls{alpha}.
5. Referencing (Using) Entries

Next use: \gls{sample} and \gls{alpha}.

\printunsrtglossary
\end{document}

Example 93: References in section headings using \glsfmttext

Contents

1 Sample and $\alpha$

Glossary

1 Sample and $\alpha$

First use: sample (an example) and $\alpha$ (alpha).

Next use: sample and $\alpha$.

Glossary

sample an example

$\alpha$ alpha

Note that this still results in “Token not allowed in a PDF string” warnings from hyperref. This is due to the maths shift and $\alpha$, and is something that would also occur if the section title explicitly contained $\alpha$. If this is likely to happen, the issue can be solved by placing \textorpdfstring within the field value. For example:

\glsnoexpandfields
\newglossaryentry{alpha}{description={alpha}, name={\textorpdfstring{$\alpha$}{alpha}}}

Note the need to prevent field expansion with \glsnoexpandfields, otherwise \textorpdfstring will be prematurely expanded while the entry is being defined.

The options noindex and hyper=false are hard-coded when the commands listed below, such as \glsfmtshort, occur in the header or contents, but within the actual section title or
5. Referencing (Using) Entries

caption in the document text, those options are obtained from:

\glsxtrtitleopts

This simply expands to the option list. For example, you may actually want a hyperlink and indexing to occur in the document body, in which case redefine \glsxtrtitleopts to do nothing:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\pagestyle{headings}
\glssetcategoryattribute{general}{headuc}{true}
\glsdefpostlink{general}{\glsxtrpostlinkAddDescOnFirstUse}
\newglossaryentry{sample}{name={sample},description={an example}}
\renewcommand{\glsxtrtitleopts}{}
\begin{document}
\section{\Glsfmttext{sample}}
First use: \gls{sample}.
Next use: \gls{sample}.
\printunsrtglossary
\end{document}

Example 94: Reference with hyperlink in section headings

1  SAMPLE

1  Sample

First use: sample (an example). Next use: sample.

Glossary

sample  an example
5. Referencing (Using) Entries

\glsfmtshort\{\textit{entry-label}\}

This normally behaves like \glsxtrshort but expands to just \glsentryshort in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtshort\{\textit{entry-label}\}

This normally behaves like \Glsxtrshort but expands to:

\MFUsentencecase\{\glsentryshort\{\textit{entry-label}\}\}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtshort\{\textit{entry-label}\}

This normally behaves like \GLSxtrshort but expands to just \glsentryshort in PDF bookmarks and is adjusted when appearing in the header or contents.

\pglsfmtshort\[\textit{options}\]\{\textit{entry-label}\}\[(\textit{insert})\] modifiers: * + \{alt-mod\}

As \glsfmtshort but inserts the \texttt{prefix} field and separator, if the \texttt{prefix} value is set and non-empty. Provided for use with \texttt{glossaries}-\texttt{prefix}.

\Pglsfmtshort\[\textit{options}\]\{\textit{entry-label}\}\[(\textit{insert})\] modifiers: * + \{alt-mod\}

As \pglsfmtshort but sentence case. Note the initial “P” in the command name, which matches \texttt{Pgls} (similarly for the other prefix sentence case commands).

\PGLSfmtshort\[\textit{options}\]\{\textit{entry-label}\}\[(\textit{insert})\] modifiers: * + \{alt-mod\}

As \pglsfmtshort but all caps.

\glsfmtshortpl\{\textit{entry-label}\}

This normally behaves like \glsxtrshortpl but expands to just \glsentryshortpl in PDF bookmarks and is adjusted when appearing in the header or contents.
5. Referencing (Using) Entries

bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtshortpl{⟨entry-label⟩}

This normally behaves like \Glsxtrshortpl but expands to:

\MFUsentencecase{\glsentryshortpl{⟨entry-label⟩}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtshortpl{⟨entry-label⟩}

This normally behaves like \GLSxtrshortpl but expands to just \glsentryshortpl in PDF bookmarks and is adjusted when appearing in the header or contents.

\pglsfmtshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩

As \glsfmtshortpl but inserts the prefixplural field and separator, if the prefixplural value is set and non-empty. Provided for use with glossaries-prefix.

\Pglsfmtshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩

As \pglsfmtshortpl but sentence case.

\PGLSfmtshortpl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩

As \pglsfmtshortpl but all caps.

\glsfmtlong{⟨entry-label⟩}

This normally behaves like \glsxtrlong but expands to just \glsentrylong in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glslong{⟨entry-label⟩}

This normally behaves like \Glssxtrlong but expands to:
5. Referencing (Using) Entries

\MFUsentencecase{\glsentrylong{(entry-label)}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtlong{(entry-label)}

This normally behaves like \GLSxtrlong but expands to just \glsentrylong in PDF bookmarks and is adjusted when appearing in the header or contents.

\pglsfmtlong[{\langle options\rangle}]{(entry-label)}[{\langle insert\rangle}] modifiers: * + {\langle alt-mod\rangle}

As \glsfmtlong but inserts the prefix\texttt{first} field and separator, if the prefix\texttt{first} value is set and non-empty. Provided for use with glossaries--prefix.

\PGLSfmtlong[{\langle options\rangle}]{(entry-label)}[{\langle insert\rangle}] modifiers: * + {\langle alt-mod\rangle}

As \pglsfmtlong but sentence case.

\PGLSfmtlongpl[{\langle options\rangle}]{(entry-label)}[{\langle insert\rangle}] modifiers: * + {\langle alt-mod\rangle}

As \pglsfmtlong but all caps.

\glsfmtlongpl{(entry-label)}

This normally behaves like \glsxtrlongpl but expands to just \glsentrylongpl in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtlongpl{(entry-label)}

This normally behaves like \GLSxtrlongpl but expands to:

\MFUsentencecase{\glsentrylongpl{(entry-label)}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtlongpl{(entry-label)}
5. Referencing (Using) Entries

This normally behaves like \GLSxtrlongpl but expands to just \glsentrylongpl in PDF bookmarks and is adjusted when appearing in the header or contents.

\pglsfmtlongpl[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As \pglsfmtlongpl but inserts the prefixfirstplural field and separator, if the prefix-firstplural value is set and non-empty. Provided for use with glossaries-prefix.

\Pglssfmtlongpl[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As \pglssfmtlongpl but sentence case.

\PGLSfmtlongpl[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

As \pglssfmtlongpl but all caps.

The full form is slightly different as it doesn’t correspond to an individual field but instead is formed from a combination of the short and long fields (the order depending on the abbreviation style). Since it’s too complicated to simply expand to the appropriate style, a simple expandable command is provided for the PDF bookmarks:

\glspdffmffull{⟨entry-label⟩}

This just expands to the long form followed by the short form in parentheses:

\newcommand*{\glspdffmffull}[1]{\glsentrylong{#1} (\glsentryshort{#1})}

You will need to redefine this if you require the short form first. There is an analogous command for the plural:

\glspdffmffullpl{⟨entry-label⟩}

This has a similar definition to \glspdffmffull but uses \glsentrylongpl and \glsentryshortpl.
5. Referencing (Using) Entries

This normally behaves like \glsxtrfull but expands to: just \glspdffmtfull in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtfull\{⟨entry-label⟩\}

This normally behaves like \Glsxtrfull but expands to:

\MFUsentencecase{\glspdffmtfull\{⟨entry-label⟩\}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtfull\{⟨entry-label⟩\}

This normally behaves like \GLSxtrfull but expands to just \glspdffmtfull in PDF bookmarks and is adjusted when appearing in the header or contents.

\glsfmtfullpl\{⟨entry-label⟩\}

This normally behaves like \glsxtrfullpl but expands to just \glspdffmtfullpl in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtfullpl\{⟨entry-label⟩\}

This normally behaves like \GLSxtrfullpl but expands to just \glspdffmtfullpl in PDF bookmarks and is adjusted when appearing in the header or contents.

\glsfmtname\{⟨entry-label⟩\}

This normally behaves like \glsxtrfullpl but expands to:

\MFUsentencecase{\glspdffmtfullpl\{⟨entry-label⟩\}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtfullpl\{⟨entry-label⟩\}

This normally behaves like \GLSxtrfull but expands to just \glspdffmtfull in PDF bookmarks and is adjusted when appearing in the header or contents.

\glsfmtname\{⟨entry-label⟩\}
This normally behaves like \texttt{\glsname} but expands to just \texttt{\glsentryname} in PDF bookmarks and is adjusted when appearing in the header or contents.

\texttt{\Glsfmtname{⟨entry-label⟩}}

This normally behaves like \texttt{\Glsname} but expands to:

\texttt{\MFUsentencecase{\glsentryname{⟨entry-label⟩}}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\texttt{\GLSfmtname{⟨entry-label⟩}}

This normally behaves like \texttt{\GLSname} but expands to just \texttt{\glsentryname} in PDF bookmarks and is adjusted when appearing in the header or contents.

\texttt{\glsfmttext{⟨entry-label⟩}}

This normally behaves like \texttt{\glstext} but expands to just \texttt{\glsentrytext} in PDF bookmarks and is adjusted when appearing in the header or contents.

\texttt{\Glsfmttext{⟨entry-label⟩}}

This normally behaves like \texttt{\Glstext} but expands to:

\texttt{\MFUsentencecase{\glsentrytext{⟨entry-label⟩}}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\texttt{\GLSfmttext{⟨entry-label⟩}}

This normally behaves like \texttt{\GLStext} but expands to just \texttt{\glsentrytext} in PDF bookmarks and is adjusted when appearing in the header or contents.

\texttt{\glsfmtplural{⟨entry-label⟩}}
5. Referencing (Using) Entries

This normally behaves like `\glsplural` but expands to just `\glsentryplural` in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtplural{⟨entry-label⟩}

This normally behaves like `\Glsplural` but expands to:

\MFUsentencecase{\glsentryplural{⟨entry-label⟩}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtplural{⟨entry-label⟩}

This normally behaves like `\GLSplural` but expands to just `\glsentryplural` in PDF bookmarks and is adjusted when appearing in the header or contents.

\glsfmtfirst{⟨entry-label⟩}

This normally behaves like `\glsfirst` but expands to just `\glsentryfirst` in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtfirst{⟨entry-label⟩}

This normally behaves like `\Glsfirst` but expands to:

\MFUsentencecase{\glsentryfirst{⟨entry-label⟩}}

in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtfirst{⟨entry-label⟩}

This normally behaves like `\GLSfirst` but expands to just `\glsentryfirst` in PDF bookmarks and is adjusted when appearing in the header or contents.

\glsfmtfirstpl{⟨entry-label⟩}
This normally behaves like \glsfirstplural but expands to just \glsentryfirstplural in PDF bookmarks and is adjusted when appearing in the header or contents.

\Glsfmtfirstpl{⟨entry-label⟩}

This normally behaves like \Glsfirstplural but expands to:

\MFUsentencecase{\glsentryfirstplural{⟨entry-label⟩}}
in PDF bookmarks and is adjusted when appearing in the header or contents.

\GLSfmtfirstpl{⟨entry-label⟩}

This normally behaves like \GLSfirstplural but expands to just \glsentryfirstplural in PDF bookmarks and is adjusted when appearing in the header or contents.

5.3.3. Advanced Commands

This section is intended for advanced users and package developers.

The commands described here are irrelevant if you use \glsxtrRevertMarks to restore the definitions of \markright, \markboth and \@starttoc. If you use \glsxtrRevertTocMarks, then this section is only applicable to \markright and \markboth.

If you need to know whether or not some code is inside a header or contents list, you can use:

\glsxtrifinmark{⟨true⟩}{⟨false⟩}

This does ⟨true⟩ if the command occurs within \markright, \markboth or \@starttoc otherwise does ⟨false⟩.

If you need to know whether or not some code is inside a contents list (but not the header), you can use:

\glsxtrifintoc{⟨true⟩}{⟨false⟩}

This does ⟨true⟩ if the command occurs within \@starttoc otherwise it does ⟨false⟩. (The modified definition of \@starttoc sets \glsxtrifintoc to \@firstoftwo at the start and to \@secondoftwo at the end.)
5. Referencing (Using) Entries

If you need to know whether or not some code is in the PDF bookmarks or heading, you can use:

\glsxtrtitleorpdforheading{⟨title⟩}{⟨PDF bookmarks⟩}{⟨heading⟩}

This does the applicable argument depending on whether the command occurs within a title/caption or PDF bookmark or heading.

If this command occurs within the toc file, it will do its ⟨heading⟩ argument but if \glsxtrtitleorpdforheading expands while it’s being written to the toc file, then it will expand to ⟨title⟩. This can be illustrated in the following document:

\documentclass{report}
\usepackage{lipsum}
\usepackage{hyperref}
\usepackage{glossaries-extra}
\pagestyle{headings}
\begin{document}
\tableofcontents
\chapter{\glsxtrtitleorpdforheading{Title}{PDF}{Heading}\glsxtrifinmark{in mark}{not in mark}}
\lipsum
\chapter{\protect\glsxtrtitleorpdforheading{Title}{PDF}{Heading}\protect\glsxtrifinmark{in mark}{not in mark}}
\lipsum
\end{document}

In the first case, \glsxtrtitleorpdforheading expands as it's being written to the toc file, so it expands to “Title”. In the second case, \glsxtrtitleorpdforheading is protected so that command is written to the toc file. On the next \LaTeX, when the table of contents is displayed, this command will expand to “Heading”, because it's in the toc file. Similarly, in the first case, \glsxtrifinmark will expand to “not in mark” as it's written to the toc file, but in the second case it's expansion is prevented, so it will expand to “in mark” in the table of contents.

If \gettitlestring has been loaded (used by nameref to provide \nameref) then adjustments for both \glsxtrtitleorpdforheading and \glsxtrifinmark will be added to \GetTitleStringDisableCommands, but bear in mind that you will need to use the following for it to have an effect:

\GetTitleStringSetup{expand}

The commands described in §5.3.2, such as \glsfmtshort, are essentially defined as:

215
If `\textorpdfstring` isn’t defined, then the definition is:

\textorpdfstring{\glsxtrtitle{field}}{(entry-label)}

For example, `\glsfmtshort` is defined as (with `hyperref`):

\newcommand*{\glsfmtshort}{\textorpdfstring{\glsxtrtitleshort{#1}}{\glsentryshort{#1}}}

This ensures that `\glsfmtshort` expands to just `\glsentryshort` within the PDF bookmarks. Provided the field value doesn’t contain any problematic commands, this allows the actual value to be added to the bookmarks.

Unfortunately the case-changing commands can’t expand and therefore aren’t appropriate for the bookmarks (which need to be a PDF string). This means that the sentence case and all caps commands also use the unmodified field value for the bookmark. For example, `\Glsfmtshort` is defined as:

\newcommand*{\Glsfmtshort}{\textorpdfstring{\Glsxtrtitleshort{#1}}{\glsentryshort{#1}}}

The `\glsxtrtitle{field}` set of commands all default to the corresponding `\glstext`-like command with the options given by `\glsxtrtitleopts` and an empty insert final argument. These title commands are redefined by `\glsxtrmarkhook` to the corresponding `\glsxtrhead{field}` command. These “head” commands use `\NoCaseChange` to prevent interference from page headers that convert to all caps (which can inappropriately convert the entry label to all caps). Instead, the `headuc` attribute needs to be set to `true` to use the appropriate all caps command. A shortcut command is provided to test for this attribute:

\glsxtrifheaduc{(entry-label)}{(true)}{(false)}

This is defined as:

\newcommand*{\glsxtrifheaduc}{\glsxtrifintoc{#3}{\glsifattribute{#1}{headuc}{true}{#2}{#3}}}

216
5. Referencing (Using) Entries

Since the header commands also end up in the contents, where the all caps conversion should not be applied, the definition includes \glsxtrifintoc to skip the check in the contents.

\glsxtrifintoc{(entry-label)}

The normal behaviour of \glsfmtshort. This is redefined by \glsxtrmarkhook to \glsxtrheadshort. The default is:

\glsxtrshort[noindex,hyper=false]{(entry-label)}[]

(This is performed indirectly via an internal command that ensures that \glsxtrtitleopts is expanded before being passed in the optional argument.)

\glsxtrheadshort{(entry-label)}

Used to display the short form in the page header. This is defined as:

\newcommand*{\glsxtrheadshort}[1]{\%\protect\NoCaseChange{\%\glsxifattribute{#1}{headuc}{true}\%\GLSxtrshort[noindex,hyper=false]{#1}[]\%\%\glsxtrshort[noindex,hyper=false]{#1}[]\%\}%}

The sentence case commands also check the headuc attribute.

\glsxtrshort{(entry-label)}

The normal behaviour of \Glsfmtshort. This is redefined by \glsxtrmarkhook to \Glsxtrheadshort. The default is:

\Glsxtrshort[noindex,hyper=false]{(entry-label)}[]

(Again, this is performed indirectly via an internal command that ensures that \glsxtr-
5. Referencing (Using) Entries

titleopts is expanded before being passed in the optional argument.

\Glsxtrheadshort{⟨entry-label⟩}

Used to display the sentence case short form in the page header. This is defined as:

\begin{verbatim}
\newcommand*{\Glsxtrheadshort}[1]{%
  \protect\NoCaseChange
  {%
    \glsifattribute{#1}{headuc}{true}{%
      \GLSxtrshort[noindex,hyper=false]{#1}[]%
    }%
    {%
      \Glsxtrshort[noindex,hyper=false]{#1}[]%
    }%
  }%
}
\end{verbatim}

\GLSxtrtitleshort{⟨entry-label⟩}

The normal behaviour of \GLSfmtshort. This is redefined by \glsxtrmarkhook to \GLSxtrheadshort. The default uses \GLSxtrshort in a similar way to \glsxtrtitleshort and \Glsxtrtitleshort.

\GLSxtrheadshort{⟨entry-label⟩}

Used to display the all caps short form in the page header. In this case, there’s no need to check to the headuc attribute, but the label needs to be protected from any potential case-change:

\begin{verbatim}
\newcommand*{\GLSxtrheadshort}[1]{%
  \protect\NoCaseChange
  {%
    \GLSxtrshort[noindex,hyper=false]{#1}[]%
  }%
}
\end{verbatim}

All the similar commands listed below are defined in an analogous way, except for the glossaries-prefix commands, where only the sentence case title version is provided. This is
because commands like \Pglstitleshort have to determine whether or not to use \glsfmtshort or \Glsfmtshort depending on whether or not the prefix has been set. Whereas commands like \pglsfmtshort simply need to insert the prefix and separator if set and then use the corresponding \glsfmtshort.

\Pglstitleshort{(entry-label)}
The normal behaviour of \Pglstitleshort.

\Pglstitleshortpl{(entry-label)}
The normal behaviour of \Pglstitleshortpl.

\Pglstitlelong{(entry-label)}
The normal behaviour of \Pglstitlelong.

\Pglstitlelongpl{(entry-label)}
The normal behaviour of \Pglstitlelongpl.

\glsfmtshortpl{(entry-label)}
The title plural short form. (Normal behaviour of \glsfmtshortpl.)

\glsfmtshortpl{(entry-label)}
The header plural short form. (The behaviour of \glsfmtshortpl when it occurs in a header.)

\Glsfmtshortpl{(entry-label)}
The title plural sentence case short form. (Normal behaviour of \Glsfmtshortpl.)

\Glsfmtshortpl{(entry-label)}
5. Referencing (Using) Entries

The header plural sentence case short form. (The behaviour of \Glsfmtshortpl when it occurs in a header.)

\GLSxtrtitleshorth\{entry-label\}\}

The title plural all caps short form. (Normal behaviour of \GLSfmtshortpl.)

\GLSxtrheadshor\{entry-label\}\}

The header plural all caps short form. (The behaviour of \GLSfmtshortpl when it occurs in a header.)

\glsxtrtitlelong\{entry-label\}\}

The title long form. (Normal behaviour of \glsfmtlong.)

\glsxtrheadlong\{entry-label\}\}

The header long form. (The behaviour of \glsfmtlong when it occurs in a header.)

\GLSxtrtitlelong\{entry-label\}\}

The title sentence case long form. (Normal behaviour of \GLSfmtlong.)

\GLSxtrheadlong\{entry-label\}\}

The header sentence case long form. (The behaviour of \GLSfmtlong when it occurs in a header.)

\GLSxtrtitlelong\{entry-label\}\}

The title all caps long form. (Normal behaviour of \GLSfmtlong.)

\GLSxtrheadlong\{entry-label\}\}
5. Referencing (Using) Entries

The header all caps long form. (The behaviour of \GLSfmtlong when it occurs in a header.)

\glsxtrtitlelongpl\{⟨entry-label⟩\}

The title plural long form. (Normal behaviour of \glsfmtlongpl.)

\glsxtrheadlongpl\{⟨entry-label⟩\}

The header plural long form. (The behaviour of \glsfmtlongpl when it occurs in a header.)

\GLSxtrtitlelongpl\{⟨entry-label⟩\}

The title plural sentence case long form. (Normal behaviour of \GLSfmtlongpl.)

\GLSxtrheadlongpl\{⟨entry-label⟩\}

The header plural sentence case long form. (The behaviour of \GLSfmtlongpl when it occurs in a header.)

\GLSxtrtitlelongpl\{⟨entry-label⟩\}

The title plural all caps long form. (Normal behaviour of \GLSfmtlongpl.)

\GLSxtrheadlongpl\{⟨entry-label⟩\}

The header plural all caps long form. (The behaviour of \GLSfmtlongpl when it occurs in a header.)

\glsxtrtitlefull\{⟨entry-label⟩\}

The title full form. (Normal behaviour of \glsfmtfull.)

\glsxtrheadfull\{⟨entry-label⟩\}

The header full form. (Normal behaviour of \glsfmtfull.)
5. Referencing (Using) Entries

The header full form. (The behaviour of \texttt{\textbackslash glsfmtfull} when it occurs in a header.)

\texttt{\GLSxtrtitlefull\{}\langle entry-label\rangle\}\texttt{\}}

The title sentence case full form. (Normal behaviour of \texttt{\GLSfmtfull}.)

\texttt{\GLSxtrheadfull\{}\langle entry-label\rangle\}\texttt{\}}

The header sentence case full form. (The behaviour of \texttt{\GLSfmtfull} when it occurs in a header.)

\texttt{\GLSxtrtitlefull\{}\langle entry-label\rangle\}\texttt{\}}

The title all caps full form. (Normal behaviour of \texttt{\GLSfmtfull}.)

\texttt{\GLSxtrheadfull\{}\langle entry-label\rangle\}\texttt{\}}

The header all caps full form. (The behaviour of \texttt{\GLSfmtfull} when it occurs in a header.)

\texttt{\glsxtrtitlefullpl\{}\langle entry-label\rangle\}\texttt{\}}

The title plural full form. (Normal behaviour of \texttt{\glsfmtfullpl}.)

\texttt{\glsxtrheadfullpl\{}\langle entry-label\rangle\}\texttt{\}}

The header plural full form. (The behaviour of \texttt{\glsfmtfullpl} when it occurs in a header.)

\texttt{\Glsxtrtitlefullpl\{}\langle entry-label\rangle\}\texttt{\}}

The title plural sentence case full form. (Normal behaviour of \texttt{\Glsfmtfullpl}.)

\texttt{\Glsxtrheadfullpl\{}\langle entry-label\rangle\}\texttt{\}}

The header plural sentence case full form. (The behaviour of \texttt{\Glsfmtfullpl} when it occurs
5. Referencing (Using) Entries

In a header.)

\GLSxtrtitlefullpl\{⟨entry-label⟩\}

The title plural all caps full form. (Normal behaviour of \GLSfmtfullpl.)

\GLSxtrheadfullpl\{⟨entry-label⟩\}

The header plural all caps full form. (The behaviour of \GLSfmtfullpl when it occurs in a header.)

\glsxtrtitlename\{⟨entry-label⟩\}

The title name field. (Normal behaviour of \glsfmtname.)

\glsxtrheadname\{⟨entry-label⟩\}

The header name field. (The behaviour of \glsfmtname when it occurs in a header.)

\Glsxtrtitlename\{⟨entry-label⟩\}

The title sentence case name field. (Normal behaviour of \Glsfmtname.)

\Glsxtrheadname\{⟨entry-label⟩\}

The header sentence case name field. (The behaviour of \Glsfmtname when it occurs in a header.)

\GLSxtrtitlefullpl\{⟨entry-label⟩\}

The title all caps name field. (Normal behaviour of \GLSfmtname.)

\GLSxtrheadname\{⟨entry-label⟩\}
The header all caps name field. (The behaviour of \GLSfmtname when it occurs in a header.)

\glsxtrtitletext{(entry-label)}

The title text field. (Normal behaviour of \glsfmttext.)

\glsxtrheadtext{(entry-label)}

The header text field. (The behaviour of \glsfmttext when it occurs in a header.)

\GLSxtrtitletext{(entry-label)}

The title sentence case text field. (Normal behaviour of \Glssfmttext.)

\GLSxtrheadtext{(entry-label)}

The header sentence case text field. (The behaviour of \Glssfmttext when it occurs in a header.)

\GLSxtrtitletext{(entry-label)}

The title all caps text field. (Normal behaviour of \GLSfmttext.)

\GLSxtrheadtext{(entry-label)}

The header all caps text field. (The behaviour of \GLSfmttext when it occurs in a header.)

\glsxtrtitleplural{(entry-label)}

The title plural field. (Normal behaviour of \glsfmtplural.)

\glsxtrheadplural{(entry-label)}
5. Referencing (Using) Entries

The header plural field. (The behaviour of \texttt{\glsfmtplural} when it occurs in a header.)

\Glsxtrtitleplural{⟨entry-label⟩}

The title sentence case plural field. (Normal behaviour of \texttt{\Glsxtrtitleplural}.)

\Glsxtrheadplural{⟨entry-label⟩}

The header sentence case plural field. (The behaviour of \texttt{\Glsxtrheadplural} when it occurs in a header.)

\GLSxtrtitleplural{⟨entry-label⟩}

The title all caps plural field. (Normal behaviour of \texttt{\GLSfmtplural}.)

\GLSxtrheadplural{⟨entry-label⟩}

The header all caps plural field. (The behaviour of \texttt{\GLSfmtplural} when it occurs in a header.)

\glsxtrtitlefirst{⟨entry-label⟩}

The title first field. (Normal behaviour of \texttt{\glsfmtfirst}.)

\glsxtrheadfirst{⟨entry-label⟩}

The header first field. (The behaviour of \texttt{\glsfmtfirst} when it occurs in a header.)

\Glsxtrtitlefirst{⟨entry-label⟩}

The title sentence case first field. (Normal behaviour of \texttt{\Glsxtrtitlefirst}.)

\Glsxtrheadfirst{⟨entry-label⟩}

The header sentence case first field. (The behaviour of \texttt{\Glsxtrheadfirst} when it occurs in a
5. Referencing (Using) Entries

\begin{itemize}
  \item \texttt{\GLSxtrtitlefirst\{entry-label\}}
  \hspace{1cm} The title all caps \texttt{first} field. (Normal behaviour of \texttt{\GLSfmtfirst}.)
  \item \texttt{\GLSxtrheadfirst\{entry-label\}}
  \hspace{1cm} The header all caps \texttt{first} field. (The behaviour of \texttt{\GLSfmtfirst} when it occurs in a header.)
  \item \texttt{\glsxtrtitlefirstplural\{entry-label\}}
  \hspace{1cm} The title \texttt{firstplural} field. (Normal behaviour of \texttt{\Glsfmtfirstpl}.)
  \item \texttt{\glsxtrheadfirstplural\{entry-label\}}
  \hspace{1cm} The header \texttt{firstplural} field. (The behaviour of \texttt{\Glsfmtfirstpl} when it occurs in a header.)
  \item \texttt{\Glsxtrtitlefirstplural\{entry-label\}}
  \hspace{1cm} The title sentence case \texttt{firstplural} field. (Normal behaviour of \texttt{\Glsfmtfirstpl}.)
  \item \texttt{\Glsxtrheadfirstplural\{entry-label\}}
  \hspace{1cm} The header sentence case \texttt{firstplural} field. (The behaviour of \texttt{\Glsfmtfirstpl} when it occurs in a header.)
  \item \texttt{\GLSxtrtitlefirstplural\{entry-label\}}
  \hspace{1cm} The title all caps \texttt{firstplural} field. (Normal behaviour of \texttt{\GLSfmtfirstpl}.)
  \item \texttt{\GLSxtrheadfirstplural\{entry-label\}}
  \hspace{1cm} The header all caps \texttt{firstplural} field. (The behaviour of \texttt{\GLSfmtfirstpl} when it occurs in a header.)
\end{itemize}
5. Referencing (Using) Entries

The definitions of \markright, \markboth and \@starttoc are saved (using \let) when glossaries-extra loads.

\@glsxtr@org@markright\{⟨text⟩\}

The previous definition of \markright.

\@glsxtr@org@markboth\{⟨left text⟩\}{⟨right text⟩}

The previous definition of \markboth.

\@glsxtr@org@@starttoc\{⟨toc⟩\}

The previous definition of \@starttoc.

The glossaries-extra definitions of \markright, \markboth and \@starttoc all start and end with hooks that redefine commands that are sensitive to being in the header or contents.

\glsxtrmarkhook

This saves the original definitions and redefines the sensitive commands. This includes \MakeUppercase which is \let to \MakeTextUppercase.

\@glsxtrinmark

This redefines \glsxtrifinmark to just do its first argument (⟨true⟩).

\@glsxtrnotinmark

This redefines \glsxtrifinmark to just do its second argument (⟨false⟩).

\glsxtrrestoremarkhook

This restores the sensitive commands to the saved definitions. (For use where grouping will cause interference.) For example, \markboth is redefined as:
5. Referencing (Using) Entries

5.4. Nested Links

Complications arise when you use the \gls-like commands in the value of the name field (or text or first fields, if set). This tends to occur with abbreviations that extend other abbreviations. For example, SHTML is an abbreviation for SSI enabled HTML, where SSI is an abbreviation for Server Side Includes and HTML is an abbreviation for Hypertext Markup Language.

For example, things can go wrong if the following is used with the glossaries package:

\newacronym{ssi}{SSI}{Server Side Includes}
\newacronym{html}{HTML}{Hypertext Markup Language}
\newacronym{shtml}{S\gls{html}}{\gls{ssi} enabled \gls{html}}

The main problems are:

1. With older versions of mfirstuc and glossaries, the sentence case commands, such as \Gls, won’t work for the shtml entry on first use if the long form is displayed before the short form (which is the default abbreviation style). This will attempt to do

\gls{\uppercase ssi} enabled \gls{html}

which just doesn’t work. Grouping the \gls{ssi} doesn’t work either as this will effectively try to do:

\uppercase{\gls{ssi}} enabled \gls{html}

This will upper case the label ssi so the entry won’t be recognised. This problem will also occur if you use the all caps version, such as \GLS{shtml}.

With mfirstuc v2.08+ and glossaries v1.49+, this issue should now be resolved for sentence case where \gls{ssi} will be mapped to \Gls{ssi} within \Gls{shtml}. The all caps command \GLS{shtml} will treat \gls as an exclusion and so won’t perform a case-change. See §5.2 for further details.

228
5. Referencing (Using) Entries

2. The long and abbreviated forms accessed through \glsentrylong and \glsentryshort are no longer expandable and so can’t be used be used in contexts that require this, such as PDF bookmarks.

3. The nested commands may end up in the sort key, which will confuse the indexing.

4. The \shtml entry produces inconsistent results depending on whether the ssi or html entries have been used. Suppose both ssi and html are used before shtml. For example:

   In the above, the first use of the shtml entry produces “SSI enabled HTML (SHTML)”. Now let’s suppose the html entry is used before the shtml but the ssi entry is used after the shtml entry, for example:

   In this case, the first use of the shtml entry now produces “server side includes (SSI) enabled HTML (SHTML)”, which looks a bit cumbersome. Now let’s suppose the shtml entry is used before (or without) the other two entries:

   Now the first use of the shtml entry produces “server side includes (SSI) enabled hypertext markup language (HTML) (SHTML)”, which looks strange.

   This is all aggravated when using just the base glossaries package when the acronym style is set with \setacronymstyle. For example:

   \setacronymstyle{long-short}
as this references the label through the use of \glslabel when displaying the long and short forms, but this value changes with each use of \gls, so instead of displaying "(SHTML)" at the end of the first use, it now displays "(HTML)", since \glslabel has been changed to html by \gls{html}.

In v1.48, the glossaries-extra package added grouping with \glslinkwrcntent, which scoped the link text. Unfortunately this grouping caused problems in math mode and had to be removed in v1.49. You can redefine \glslinkwrcntent to put the grouping back, but it still won’t scope the definitions of the placeholder commands, such as \glslabel, which need to be outside of this scope for the benefit of the post-link hook.

Another oddity occurs if you reset the html entry between uses of the shtml entry. For example:

\gls{shtml} ... \glsreset{html}\gls{shtml}

The next use of shtml produces “Shypertext markup language (HTML)”, which is downright weird. (This is a result of the short form being set to S\gls{html}, but \gls{html} is showing the full form.)

Even without this, the short form has nested formatting commands, which amount to \acronymfont{S\acronymfont{HTML}}. This may not be a problem for some styles, but if you use one of the “sm” styles (that use \textsmaller), this will produce an odd result.

5. Each time the shtml entry is used, the html entry will also be indexed and marked as used, and on first use this will happen to both the ssi and html entries. This kind of duplication in the location list isn’t usually particularly helpful to the reader.

6. If hyperref is in use, you’ll get nested hyperlinks and there’s no consistent way of dealing with this across the available PDF viewers. If on the first use case, the user clicks on the "HTML" part of the “SSI enabled HTML (SHTML)” link, they may be directed to the HTML entry in the glossary or they may be directed to the SHTML entry in the glossary.

For these reasons, with just the base glossaries package, it’s better to use the simple expandable commands like \glsentrytext or \glsentryshort in the definition of other entries. The glossaries-extra package provides two other ways of dealing with these problems:

1. If the term can simply be treated as a series of previously defined entries, then consider using multi-entries (or compound sets), as described in §7. This deals with all the issues, including case-changing.

2. Use the partially-expandable \glsxtrp, described below.
5. Referencing (Using) Entries

\texttt{\glsxtrp\{\langle field\rangle\}\{\langle entry-label\rangle\}}

where \(\langle field\rangle\) is the internal field label.

This command partially expands, so it will expand to just the field value if it occurs in the PDF bookmarks. Otherwise it will behave much like the commands described in §5.3.2, but with additional outer scoping and the post-link hook is suppressed.

Rather than testing the existence of the given field, this tests the existence of \texttt{\glss\{field\}} or \texttt{\glsxtr\{field\}}, which means that it may be confused if the \(\langle field\rangle\) argument is set to something that isn’t a field but happens to match either of those command names (such as full).

The post-link hook is suppressed by the initialisation command:

\texttt{\glsxtrpInit\{\langle cs-name\rangle\}\{\langle entry-label\rangle\}}

This is used inside the added outer scoping and is simply defined as:

\texttt{\newcommand\glsxtrpInit[2]{\let\glspostlinkhook\relax}}

It is possible to redefine this command to allow the \texttt{\glspostlinkhook} to be used, but any look-ahead (such as checking for a following punctuation character) won’t work because of the added grouping. The arguments are ignored by default. If you want to redefine \texttt{\glsxtrpInit} the first argument is the name of the control sequence that will be used, without the leading backslash (for example, \texttt{glstext} or \texttt{glsxtrshort}) and the second argument is the entry’s label.

Note that, as with commands like \texttt{\glsfmtshort}, there’s no optional argument. The default settings are \texttt{noindex} and \texttt{hyper=false}. You can change this with:

\texttt{\glsxtrsetpopts\{\langle options\rangle\}}

The argument should be the new default options.

At the start of each glossary, the default options are locally changed with:

\texttt{\glossxtrsetpopts}

This is defined as:

\texttt{\newcommand*{\glossxtrsetpopts}{{\%\glssxtrsetpopts\{noindex\}\%}}}
This allows hyperlinks for any instance of \glsxtrp{field} that occurs in the name or description, where it shouldn’t be problematic.

There are also sentence case and all caps versions.

\Glsxtrp{(field)}{(entry-label)}

This uses the corresponding sentence case command \Gls{field} or \Glsxtr{field}.

\GLSxtrp{(field)}{(entry-label)}

This uses the corresponding all caps command \GLS{field} or \GLSxtr{field}.

There are some shortcut commands for the most common fields:

\glsps{(entry-label)}

which is equivalent to \glsxtrp{short}{(entry-label)}, and

\glspt{(entry-label)}

which is equivalent to \glsxtrp{text}{(entry-label)}. As well as sentence case and all caps versions:

\GLSps{(entry-label)}

which is equivalent to \GLSxtrp{short}{(entry-label)};

\GLSpt{(entry-label)}

which is equivalent to \GLSxtrp{text}{(entry-label)};

\GLSPs{(entry-label)}

which is equivalent to \GLSPxtrp{short}{(entry-label)}, and

\GLSPt{(entry-label)}

232
which is equivalent to $\text{\GLSxtrp\{text\}}{(entry-label)}$.

For example:

```latex
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\setabbreviationstyle{long-em-short-em}
\newabbreviation{html}{HTML}{hypertext markup language}
\newabbreviation{ssi}{SSI}{server-side includes}
\newabbreviation{shtml}{SHTML}{\text{\GLSps\{html\} enabled \GLSps\{ssi\}}}
\begin{document}
\tableofcontents
\section{\GLSfmtlong{shtml}}
First use: \text{\gls{shtml}}, \text{\gls{html}}, \text{\gls{ssi}}.
Next use: \text{\gls{shtml}}, \text{\gls{html}}, \text{\gls{ssi}}.

\printunsrtglossaries
\end{document}
```
Example 95: Nested link text with \glspl

Contents

1 HTML enabled SSI 1

Glossary

1 HTML enabled SSI

First use: HTML enabled SSI (SHTML), hypertext markup language (HTML), server-side includes (SSI).

Next use: SHTML, HTML, SSI.

Glossary

HTML hypertext markup language

SSI server-side includes

SHTML HTML enabled SSI

The way that this works is as follows:

- \glsfmtlong{shtml} expands to \glsentrylong{shtml} within the PDF bookmarks, which expands to the value of the long field:

  \glsp{html\ enabled \ glsp{ssi}}

This means that \glsp (within the PDF bookmarks) in turn expands to \glsentryshort. So the bookmark text (which can’t contain any formatting commands) ends up as “HTML enabled SSI”.

- \glsfmtlong{shtml} essentially behaves like \glxsstrlong, but with the indexing and hyperlink suppressed. The link text is the value of the long field encapsulated with the abbreviation style’s formatting command (\glslongemfont in this case):

  \glslongemfont{\glsp{html\ enabled \ glsp{ssi}}}

This then becomes:
5. Referencing (Using) Entries

\begin{quote}
\textbackslash gls\textlongfont\{\textbackslash let\textbackslash glspostlinkhook\textbackslash relax
\textbackslash gls\textxshortfont\{html\}\} \text{ enabled }
\{\textbackslash let\textbackslash glspostlinkhook\textbackslash relax
\textbackslash gls\textxshortfont\{ssi\}\}\}
\end{quote}

Note the grouping and localised suppression of the post-link hook.

Note that in the above example, with older versions of \texttt{mfirstuc} and \texttt{glossaries}, it’s not possible to use \Glsxtrlong{shtml} or similar. The problem here is that it will attempt to do:

\begin{quote}
\textbackslash m\text{akefirstuc}\{\textbackslash gls\text{ps}\{html\}\text{ enabled }\textbackslash gls\text{ps}\{ssi\}\}
\end{quote}

This will essentially end up as:

\begin{quote}
\textbackslash gls\text{ps}\{\textbackslash uppercase \textbackslash html\} \text{ enabled } \textbackslash gls\text{ps}\{ssi\}
\end{quote}

which doesn’t work. If you want to protect against automated case-changes, such as using the \texttt{glossdesc} attribute, insert an empty brace at the start:

\begin{quote}
\texttt{\newabbreviation\{shtml\}\{SHTML\}\{}	extbackslash gls\text{ps}\{html\}\text{ enabled }
\textbackslash gls\text{ps}\{ssi\}\}
\end{quote}

Alternatively, upgrade to \texttt{mfirstuc v2.08+} and \texttt{glossaries v4.50+}. See §5.2.

5.5. Adjusting the Text Style

The \texttt{\textbackslash gls\}-like and \texttt{\textbackslash glstext\}-like commands produce text that’s essentially formatted either as \texttt{(hyperoutside=true)}:

\begin{quote}
\textbackslash \texttt{(hyper-cs)}\{\langle \texttt{target} \rangle\}\{\langle \texttt{textformat-cs} \rangle\{\langle \texttt{content} \rangle\}\}\{\langle \texttt{post-link hook} \rangle\}
\end{quote}

or \texttt{(hyperoutside=false)}:

\begin{quote}
\texttt{(textformat-cs)}\{\langle \texttt{hyper-cs} \rangle\{\langle \texttt{target} \rangle\}\{\langle \texttt{content} \rangle\}\}\{\langle \texttt{post-link hook} \rangle\}
\end{quote}

If hyperlinks are enabled then \texttt{(hyper-cs)} creates the hyperlink based on \texttt{\langle target \rangle} with the hyperlink text given by the second argument. If hyperlinks aren’t enabled then \texttt{(hyper-cs)} ignores the \texttt{\langle target \rangle} argument and simply does the second argument.

The \texttt{\langle content \rangle} part is the link text, which includes the final optional \texttt{\langle insert \rangle} (if supplied). The actual content depends on the command used (for example, \texttt{\textbackslash gls} or \texttt{\textbackslash glstext}). The \texttt{\textbackslash gls\}-like commands all use the entry display style associated with the entry’s glossary type, (see §5.5.5). The \texttt{\textbackslash glstext\}-like commands set the \texttt{\langle content \rangle} to the corresponding field value.
5. Referencing (Using) Entries

with the insert appended, all encapsulated with the inner formatting (see §5.5.3), with appropriate case-changing, if required.

The abbreviation commands (\glsxtrshort, \glsxtrlong, \glsxtrfull etc) are considered part of the set of \glstext-like commands, but the content is set according to the abbreviation style (see §4.5).

The commands \glsdisp and \glslink both have the content part explicitly set in their final argument. There’s no insert optional argument as it can simply be included in the content part. The difference between them is that \glsdisp is considered a \gls-like command (it unsets the first use flag, §5.10, and uses the entry display style, §5.5.5), whereas \glslink is considered a \glstext-like command.

The \text{post-link hook} part is described in §5.5.4.

The \text{textformat-cs} command is the outer formatting command, described in §5.5.1. This doesn’t include the post-link hook. If you want to include the post-link hook then you need to encapsulate the entire \gls-like and \glstext-like command (including the final optional argument, if present, and following punctuation, if the post-link hook looks ahead for punctuation).

Some sensitive formatting commands need to have the actual text in their argument (or else have the argument in an unbreakable box). The \text{content} part is usually too complicated for these commands. To help support this type of command, there is also an inner format, which is described in §5.5.3. In general, unless you require one of these sensitive commands, avoid setting the inner text format as it requires support from the underlying style (either the entry format style or the abbreviation style), which may not be available.

The following example document is ugly, but demonstrates the outer formatting (typewriter font), middle formatting (bold for regular entries and italic for abbreviations), inner formatting (highlighted in yellow), hyperlinks (red), and the category post-link hook (the description follows in parentheses for general entries on first use).

\usepackage{courier}
\usepackage[T1]{fontenc}
\usepackage{xcolor}
\usepackage{soul}
\usepackage[colorlinks]{hyperref}
\usepackage[nogroupskip]{glossaries-extra}
% outer formatting:
\renewcommand{\glstextformat}[1]{\texttt{#1}}
% middle formatting:
\renewcommand{\glsxtrregularfont}[1]{\textbf{#1}}
\renewcommand{\glsxtrabbreviationfont}[1]{\textit{#1}}
% inner formatting:
\renewcommand{\glsxtrdefaultentrytextfmt}[1]{\hl{#1}}
% post-link hook for 'general' category:
\glsdefpostlink{general}{\glsxtrpostlinkAddDescOnFirstUse}
5. Referencing (Using) Entries

\% define entries:
\newglossaryentry{sample}{name={sample},description={an example}}
\newabbreviation{html}{HTML}{hypertext markup language}
\newacronym{nasa}{NASA}{National Aeronautics and Space Administration}
\begin{document}
First use: \gls{sample}, \gls{html}, \gls{nasa}.

Next use: \gls{sample}, \gls{html}, \gls{nasa}.
\end{document}

This produces:

Example 96: Link text styles: outer, middle, inner, hyperlinks and post-link hook

First use: \textbf{sample} (an example), \textit{hypertext markup language} (HTML), \texttt{NASA}.

Next use: \textbf{sample}, \textit{HTML}, \texttt{NASA}.

Glossary

\textbf{sample} an example

\textbf{HTML} hypertext markup language

\texttt{NASA} National Aeronautics and Space Administration

Note that the hyperlink, outer and middle formatting aren’t applied to the post-link hook. The acronym category has the short-nolong abbreviation style, which sets the regular attribute to true. This means that the NASA entry uses the regular middle format (\glsxtr-regularfont) not the abbreviation middle format (\glsxtrabbreviationfont).

If you have a formatting command that needs to have its argument fully-expanded before being applied, you may be able to use:

\GlssXtrExpandedFmt{⟨cs⟩}{⟨content⟩}

This fully-expands ⟨content⟩ and does ⟨cs⟩⟨expanded-content⟩, where ⟨cs⟩ is a command that takes a single argument. For example, to use soul’s underlining command \ul:
5. Referencing (Using) Entries

\renewcommand{\glsxtrregularfont}[1]{\GlsXtrExpandedFmt{#1}}

(See Example 117 on page 293.) This isn’t guaranteed to work as the link text may contain fragile content.

The inner formatting can be unpredictable. For example, abbreviation styles are complicated and so the inner formatting command is included in some of the field values, such as the name, which is why the abbreviation name is highlighted in the glossary. In the above example, the inner formatting is included in the category post-link hook, but only because \glstextrpostlinkAddDescOnFirstUse is designed to include it. If the category post-link hook was simply defined as:

\glsdefpostlink{general}{%\glsxtrifwasfirstuse{ (\glsentrydesc{\glslabel})}}{}

then the inner formatting won’t be applied, since it’s not included in the hook. This is demonstrated in a slightly modified version of the above document (initial part of preamble that deals with loading packages and redefining formatting commands as before):

% post-link hook for 'general' category:
\glsdefpostlink{general}{%\glsxtrifwasfirstuse{ (\glsentrydesc{\glslabel})}}{}
% this style sets the post-link hook for 'abbreviation' category:
\setabbreviationstyle{long-postshort-user}
% this style sets the post-link hook for 'acronym' category:
\setabbreviationstyle[acronym]{short-postfootnote}
% define entries:
\newglossaryentry{sample}{name={sample},description={an example}}
\newabbreviation{html}{HTML}{hypertext markup language}
\newacronym{nasa}{NASA}{National Aeronautics and Space Administration}
\begin{document}
\Firstuse: \gls{sample}, \gls{html}, \gls{nasa}.
\Nextuse: \gls{sample}, \gls{html}, \gls{nasa}.
\end{document}

This produces:
5. Referencing (Using) Entries

Example 97: Link text styles: outer, middle, inner, hyperlinks and post-link hooks (custom and abbreviation style)

Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample</td>
<td>an example</td>
</tr>
<tr>
<td>HTML</td>
<td>hypertext markup language</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
</tbody>
</table>

The “post” abbreviation styles put some content into the post-link hook and provide support for the inner formatting. The above example sets the abbreviation style to **long-postshort-user**. This sets up the post-link hook for the associated category (**abbreviation**, in this case) to show the parenthetical material. Be aware that this will override any previous definition of that hook. This style supports the inner formatting (so the parenthetical material is highlighted).

Similarly, the **short-postfootnote** style is applied to the **acronym** category, and sets the post-link hook for that category (which looks head for punctuation). The inner formatting is applied to the footnote text but not the marker.

The post-link hook for the **general** category is now much simpler and doesn’t include support for the inner formatting, so it’s not highlighted.

None of the post-link content is incorporated into the hyperlink, outer or middle formatting.

In general, it’s better to adjust the abbreviation’s style commands (see §4.5.1.3) rather than use the middle or inner formatting if abbreviations need to be displayed in a particular font.

5.5.1. Outer Formatting

By default, the outer formatting is produced with \glisttextformat, which is defined by the base glossaries package. However it can be replaced by the **textformat** category attribute or
by the `textformat` option. The order of precedence (not cumulative) is: the option supplied to the \gls-like or \glstext-like command, the category attribute, \glstextformat. For example:

```latex
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\glsdefpostlink{general}{\(\langle\text{\glsentrydesc{\glslabel}}\rangle\)}
\newglossaryentry{sample}{name={sample},description={an example}}
\newcommand{\strong}[1]{\textbf{\color{green}#1}}
\renewcommand{\glstextformat}[1]{\emph{#1}}
\begin{document}
\gls{sample}[-insert]. \strong{\gls{sample}[-insert]}.
\glssetcategoryattribute{general}{textformat}{strong}
\gls{sample}[-insert]. \gls{hyperoutside=false}{sample}[-insert].
\gls[textformat=textsf]{sample}[-insert].
\end{document}
```

Example 98: Changing the outer text format

```
sample-insert (an example). sample-insert (an example).
sample-insert (an example). sample-insert (an example).
sample-insert (an example).
```

The red text colour is from the hyperlink (red is the default with hyperref’s `colorlinks` option). The green from the custom `\strong` command is cancelled by the hyperlink colour change when the hyperlink is inside `\strong`.

After the `textformat` attribute is set, the `\glstextformat` command isn’t used, which is why the remaining lines don’t have any italic. The final line uses the `textformat` option, which overrides the `textformat` attribute, so neither `\glstextformat` nor the custom `\strong` are used.

Note that the only time that the post-link hook is included in the formatting is when the entire `\gls` command has been encapsulated.

### 5.5.2. Middle Formatting

The middle formatting comes between the outer formatting (§5.5.1 above) and the inner formatting (§5.5.3 below).

The middle formatting is implemented by the entry format style (§5.5.5) for the \gls-like commands or is initialised by `\glsxtrassignfieldfont` for the \glstext-like commands.
5. Referencing (Using) Entries

If you provide your own custom entry format style you will need to add support for the middle formatting, if required.

\glsxtrregularfont{⟨text⟩}

The command to use for regular entries. This is initialised to just do its argument.

\glsxtrabbreviationfont{⟨text⟩}

The command to use for abbreviations that considered non-regular entries.

The following document has a regular entry (sample), a regular abbreviation (radar, which uses short–nolong the default acronym style), and a non-regular abbreviation (HTML, which uses long–short the default abbreviation style):

\newglossaryentry{sample}{name={sample},description={an example}}
\newabbreviation{html}{HTML}{hypertext markup language}
\newacronym{radar}{radar}{radio detection and ranging}
\renewcommand{\glsxtrregularfont}[1]{\emph{#1}}
\renewcommand{\glsxtrabbreviationfont}[1]{\textbf{#1}}
\begin{document}
\gls{sample}, \gls{html}, \gls{radar}.
\end{document}

Example 99: Middle formatting

\textit{sample, hypertext markup language (HTML), radar.}

Note that even though radar is an abbreviation, it’s considered a regular entry because it uses a regular style.

\glsxtrassignfieldfont{⟨entry-label⟩}

This command is used by all the \glstext-like commands to initialise the internal command used to encapsulate the field value. This will either be set to \glsxtrregularfont (for regular entries) or \@firstofone otherwise.

Note that this doesn’t use \glsxtrabbreviationfont as non-regular abbreviations are too complicated to work with \glstext, \glsfirst, \glsplural, \glsfirstplural or their case-changing variants. Instead, use the \gls-like commands or the abbreviation commands, such as \glsxtrshort.
5. Referencing (Using) Entries

5.5.3. Inner Formatting

If you want to format the link text, the best method is to either use the outer formatting or encapsulate the entire \textit{like} command, as described in §5.5.1. However, there are some sensitive commands that don’t work if the command argument doesn’t simply contain text.

Sometimes the issue may occur when the sensitive command that needs to be encapsulate \textit{like} doesn’t like boolean variables being changed (which occurs when the first use flag is unset). If this is the case, you may want to consider buffering as an alternative (see §5.10.1).

For example, if the sample document from §5.5.1 is adjusted to include the soul package and the following line is added to the document:

\gls[\textformat=hl]{sample}

then the document build will fail with the error:

! Package soul Error: Reconstruction failed.

Once solution is to do the following instead:

\hl{\mbox{\gls{sample}}}

This will now work, but the box will prevent hyphenation, so it’s only useful if the link text is short, such as a symbol. If the link text is long (such as a phrase or the first use of an abbreviation), this method can produce undesirable results with overfull or underfull lines.

The inner formatting is designed to provide a workaround, but it must be implemented deep within the entry style formatting. This means that if you provide your own custom style, you will need to add the appropriate commands if you want that style to support inner formatting. You may also need to switch to using \textit{MFU}sentencecase instead of \textit{make}-\textit{firstuc} if any of the sentence case commands are required:

\renewcommand{\glssentencecase}{1}{\textit{MFU}sentencecase{#1}}

Although there’s no guarantee that this will work for some particularly problematic formatting commands.

With the default entry style, the above example can be changed to:
5. Referencing (Using) Entries

The inner formatting may be split up in order to move them into the arguments of internal commands, such as those used for case-changing. This can result in unwanted side-effects.

The following uses \fbox (which draws a frame around its argument) and soul’s \so (which spaces out the letters):

\% requires glossaries.sty v4.50+ and mfirstuc v2.08+
\renewcommand{\glssentencecase}[1]{\MFUsentencecase{#1}}
\newacronym{radar}{radar}{radio detection and ranging}
\begin{document}
\Gls[innertextformat=fbox]{radar}'s system\ldots
\Gls[innertextformat=so]{radar}'s system\ldots
\fbox{\Gls{radar}'s} system\ldots
\so{\mbox{\Gls{radar}'s}} system\ldots
\end{document}

Example 100: Inner formatting

Radar'\textsc{s} system\ldots
R a d a r'\textsc{s} system\ldots
Radar’s system\ldots
Radar’s system\ldots

Note the fragmentation of the inner formatting. The use of \mbox in the final line prevents an error but the letters aren’t spaced out. The only way to deal with this case is to use \glsdisp or \glslink with the text explicitly written:

\glslink{radar}{\so{Radar's}} system\ldots
The above example requires mfirstuc v2.08+.

Below are the commands used to support inner formatting.

\texttt{\glsxtrgenentrytextfmt}

This is the command that’s used to encapsulate any content that should have the inner formatting applied. It should not be redefined within the document as it’s initialised within the \gls-like and \glstext-like commands. It’s used within \glsgenentryfmt and included in the helper commands used by the predefined abbreviation styles.

Sometimes it may be necessary to include \glsxtrgenentrytextfmt within the actual field value to ensure that it’s as close as possible to the text. This is performed automatically when an entry is defined if the \texttt{encapinnerfmt} or \texttt{encapnocaseinnerfmt} attributes are set. Note that even in this case, fragmentation will occur with sentence case commands like \gls or with the insert optional argument, as in the above example with \texttt{fbox} and \texttt{so}.

\texttt{\glsxtrdefaultentrytextfmt{⟨⟨text⟩⟩}}

This is the default command that \glsxtrgenentrytextfmt will be \texttt{\let} to within the \gls-like and \glstext-like commands before their options are processed. This simply does its argument but may be redefined. (See Example 117 on page 293.)

\texttt{\glsxtrattrentrytextfmt{⟨⟨text⟩⟩}}

This command applies formatting according to whether or not the \texttt{innertextformat} attribute is set. It isn’t used by default as it should rarely be needed and increases complexity. However, if you would like to provide support for the \texttt{innertextformat} attribute, you can redefine \texttt{\glsxtrdefaultentrytextfmt} to use \texttt{\glsxtrattrenentrytextfmt}:

\texttt{\renewcommand{\glsxtrdefaultentrytextfmt}{\glsxtrattrenentrytextfmt}}

This command expects the entry label to be stored in \texttt{\glslabel} (from which it obtains the category label).

The \gls-like commands use \glsxtrgenentrytextfmt within \glsgenentryfmt for regular entries or within the abbreviation style commands for non-regular abbreviations (see
The \texttt{\textglsrefentrytextfmt} commands all essentially perform the following steps:

1. Initialise the middle formatting command \texttt{\textglsrefaccessfmt} used for encapsulating the field with \texttt{\textglsxtrassignfieldfont} (see §5.5.2).

2. If \texttt{\textglsifapplyinnerfmtfield} indicates that the field value should be encapsulated by \texttt{\textglsxtrgenentrytextfmt}, then this essentially does (or appropriate case-change equivalent):

\begin{verbatim}
\langle field-font-cs\rangle\{\textglsaccessfmt\langle field\rangle\{\langle insert\rangle\}\{\textglsxtrgenentrytextfmt\}\langle entry-label\rangle\}\{\langle internal-field\rangle\}
\end{verbatim}

otherwise it does:

\begin{verbatim}
\langle field-font-cs\rangle\{\textglsaccess\langle field\rangle\{\langle entry-label\rangle\}\textglsxtrgenentrytextfmt\{\langle insert\rangle\}\}
\end{verbatim}

(See §9 for the “access” commands.)

For example, the link text for \texttt{\textglsrefentry} is:

\begin{verbatim}
\textglsifapplyinnerfmtfield\{\langle entry-label\rangle\}{text}\%
\%\langle field-font-cs\rangle\{\textglsaccessfmttext\langle insert\rangle\}\{\textglsxtrgenentrytextfmt\}\langle entry-label\rangle\}\%
\%
\%
\langle field-font-cs\rangle\{\textglsaccess\langle entry-label\rangle\}\textglsxtrgenentrytextfmt\{\langle insert\rangle\}\%
\end{verbatim}

The \texttt{\textglsaccessfmt\langle field\rangle} commands internally use \texttt{\textglsfmtfield} to apply the inner formatting.

This determines whether or not the field identified by its internal field label for the given entry should have its value encapsulated by the inner formatting command. False indicates
that the field value already contains the inner formatting command.

```
\glsexclapplyinnerfmtfield{⟨entry-label⟩}{⟨internal-field⟩}
```

Locally adds the given field identified by its internal field label to the exclusion list for the given entry.

```
\glsfmtfield{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}{⟨internal-field⟩}
```

This command applies the formatting command ⟨cs⟩ (which takes one argument) to the entry’s field value identified by the given internal field label, including ⟨insert⟩ appended. This ensures that the internal control sequence used to store the field’s value is expanded before ⟨cs⟩ is applied.

```
\GLSfmtfield{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}{⟨internal-field⟩}
```

As above but sentence case.

```
\GLSfmtfield{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}{⟨internal-field⟩}
```

As above but all caps.

### 5.5.4. Post Link Hook

The post-link hook is a convenient way of automatically appending content after each instant of the \gls-like and \glstext-like commands. The simplest method of implementing this is with the category post-link hook, which is only applied to entries that have the given category. For example, the following will place an asterisk (*) after all entries with the default general category:

```
\glsdefpostlink{general}{*}
\newglossaryentry{sample}{name={sample},symbol={X},
  description={an example}}
\begin{document}
\Gls{sample}, \glstext{sample}, \glsdesc{sample}
and \glssymbol{sample}.
\end{document}
```
5. Referencing (Using) Entries

Example 101: Category post-link hook

Sample*, sample*, an example* and X*.

Typically, the category post-link hook is more likely to include some conditional, such as to only insert text on first use. For example, \glsxtrpostlinkAddDescOnFirstUse can be used to insert the description in parentheses after the first use.

The “post” abbreviation styles all set the category post-link hook, which will overwrite any previous definition for the abbreviation’s category.

Within the post-link hook, you can use the placeholder commands, such as \glslabel (see §5.5.5), but note that you can’t use \ifglsused to determine whether or not the entry has been used, since the post-link hook comes after the entry has been unset. Instead, use \glsxtrifwasfirstuse. Additional commands provided for use within the post-link hooks are described in this section.

The post-link hook is implemented with \glspostlinkhook, which is defined by the base glossaries package. It’s used at the end of the \gls-like and \glstext-like commands. The original base definition does nothing, but glossaries-extra redefines this:

\renewcommand*{\glspostlinkhook}{% 
  \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}% 
}%

This uses:

\glsxtrpostlinkhook

which is the main glossaries-extra post-link hook.

If you are migrating over from only using the base glossaries package and you have redefined \glspostlinkhook, consider moving your modifications to the category post-link hook or prepend to \glsxtrpostlink, as some attributes and abbreviation styles rely on the features provided by \glsxtrpostlinkhook.

The main post-link hook is defined as:
Referencing (Using) Entries

\newcommand*{\glsxtrpostlinkhook}{%
  \glsxtrdiscardperiod{\glslabel}\
  {\glsxtrpostlinkendsentence}\
  {\glsxtrifcustomdiscardperiod\
    {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}\
    {\glsxtrpostlink}}\
}\
\glsxtrdiscardperiod{\langle entry-label \rangle}{\langle discarded \rangle}{\langle no discard \rangle}{\langle token \rangle}

This checks if a following full stop needs to be discarded and does the inner post-link hook \glsxtrpostlink. Note that \glsxtrdiscardperiod and \glsxtrifperiod look ahead for a following token, so if you need to modify this command, insert your custom code at the start or add it to the category post-link hook instead.

\glsxtrdiscardperiod{\langle entry-label \rangle}{\langle discarded \rangle}{\langle no discard \rangle}{\langle token \rangle}

This discards \langle token \rangle if it’s a full stop and the entry’s category attributes indicate that a full stop should be discarded (such as discardperiod). If the punctuation character is discarded, this will then do \langle discarded \rangle, otherwise it will do \langle no discard \rangle and process \langle token \rangle as usual. If the retainfirstuseperiod attribute is set, then the following command is used to determine whether or not to discard \langle token \rangle.

\glsxtrdiscardperiodretainfirstuse{\langle entry-label \rangle}{\langle discarded \rangle}{\langle no discard \rangle}{\langle token \rangle}

This was introduced in v1.49 and is defined as:

\newcommand*{\glsxtrdiscardperiodretainfirstuse}[3]{%
  \glsxtrifwassubsequentorshort{\glsxtrperiod{\glsxtrperiod{\#2}{\#3}}{\#3}}\%
}\
\glsxtrifwasfirstuse{\langle token \rangle}

This will only discard the full stop if it follows the subsequent use of a \gls-like command or if it follows one of the \glsxtrshort set of commands. Note that this has a different effect from pre v1.49 with the \glstext-like commands, but it’s more appropriate since it’s typically only the short form that requires the period to be discarded. To restore the original behaviour:

\renewcommand*{\glsxtrdiscardperiodretainfirstuse}[3]{%
  \glsxtrifwasfirstuse{\#3}{\glsxtrperiod{\glsxtrperiod{\#2}{\#3}}{\#3}}\%
}\

If you want your own custom code to determine whether or not to check for a period
5. Referencing (Using) Entries

(instead of using known category attributes), you can redefine:

\glsxtrifcustomdiscardperiod{⟨true⟩}{⟨false⟩}
\textit{initial: ⟨false⟩}

This should expand to ⟨true⟩ if a check should be performed, otherwise it should expand to ⟨false⟩. The default definition simply does ⟨false⟩.

\glsxtrpostlinkendsentence

This is done if a full stop is discarded. If there is a category post-link hook for the entry’s category, that hook is performed (\glsxtrpostlink⟨category⟩ not \glsxtrpostlink) and the full stop is put back followed by a space factor adjustment. Otherwise, just the space factor adjustment is done.

The test to determine whether or not ⟨token⟩ is a full stop is determined by:

\glsxtrifperiod{⟨true⟩}{⟨false⟩}⟨token⟩

It may be useful to test for other punctuation characters. For example, styles such as short\text{-}postfootnote will move the footnote after certain punctuation characters.

\glsxtrifnextpunc{⟨true⟩}{⟨false⟩}

This does ⟨true⟩ if it's followed by one of the set of recognised punctuation characters, otherwise it does false. The set is initialised to .,:;?! (full stop, comma, colon, semi-colon, question mark, and exclamation mark).

A convenient way of moving code after the punctuation character is to use:

\glsxtrdopostpunc{⟨code⟩}⟨token⟩

If ⟨token⟩ is a recognised punctuation character, this will place ⟨code⟩ after the token, otherwise it will be placed before the token.

The earlier example can be adapted to put the asterisk after following punctuation:

\glsdefpostlink{general}{\glsxtrdopostpunc{*}}
\newglossaryentry{sample}{name={sample},symbol={X},
description={an example}}
\begin{document}

249
5. Referencing (Using) Entries

\texttt{\textbackslash Gls\{sample\}, \textbackslash glstext\{sample\}, (\textbackslash glsdesc\{sample\}) and \textbackslash glssymbol\{sample\}. \textbackslash end\{document\}  

Example 102: Category post-link hook with punctuation lookahead

Sample,* sample,* (an example*) and X.*

Note that the asterisk isn’t moved after the closing parenthesis. This is because that character isn’t included in the default list.

You can add additional punctuation marks with:

\texttt{\textbackslash glsxtraddpunctuationmark\{\langle \texttt{token(s)} \rangle\}}

You may list multiple characters at the same time to add a batch, but don’t add any separators (including spaces).

Note that each character must be a single token, which means a single-byte character for \texttt{pdf\LaTeX}. Multi-byte characters (UTF-8) will required a native Unicode engine (\texttt{X\LaTeX} or \texttt{Lua\LaTeX}).

For example:

\texttt{\textbackslash glsxtraddpunctuationmark\{-'\}/}

This adds three extra punctuation marks (hyphen, apostrophe and slash). Note that this doesn’t allow for closing double-quotes and will break ‘’ (double apostrophe sequence for a closing double-quote) if found. The following will only work with \texttt{X\LaTeX} or \texttt{Lua\LaTeX}:

\texttt{\usepackage\{fontspec\} \usepackage\{glossaries-extra\} \glsxtraddpunctuationmark\{"\}}

You can set the list with:

\texttt{\glsxtrsetpunctuationmarks\{\langle \texttt{token list} \rangle\}}
5. Referencing (Using) Entries

This will remove the default set as well as any additional characters. As above, each character must be a single token with no separators in the list. For example:

\glsxtrsetpunctuationmarks{.?!}

This sets the list to just three punctuation characters (so comma, colon, and semi-colon are no longer recognised).

\glsxtrpostlink

This does the category post-link hook (or nothing if it hasn’t been defined):

\newcommand*{\glsxtrpostlink}{%\csuse{glsxtrpostlink\glscategory{\glslabel}}%}

Customisation is best performed within the category post-link hook, which can be defined (or redefined) with:

\glsdefpostlink{⟨category⟩}{⟨definition⟩}

The first argument is the category label and the second is the code to perform. Note that this doesn’t check if the hook has already been defined for the category. The hook is a command in the form \glsxtrpostlink⟨category⟩. If the category label only consists of letters, you can also use \newcommand or \renewcommand instead.

\glspretopostlink{⟨category⟩}{⟨code⟩}

Similar to the above but prepends ⟨code⟩ to the associated hook (or simply defines it, if the hook doesn’t already exist).

\glsapptopostlink{⟨category⟩}{⟨code⟩}

Similar to the above but appends ⟨code⟩ to the associated hook.

Take care not to choose category labels that will cause a conflict. For example, endsentence and hook will conflict with the commands \glsxtrpostlinkend-
If you want code in the post-link hook that's not dependent on the category, consider prepending it to `\glsxtrpostlink` or `\glsxtrpostlinkhook`. Don't append it to `\glsxtrpostlinkhook` otherwise it will interfere with the punctuation lookahead.

For convenience, some commands are provided that may be of use in the category post-link hook:

- `\glsxtrpostlinkAddDescOnFirstUse` This will add the description in parentheses if the hook follows the first use of the entry. This incorporates the inner formatting and description accessibility support, if provided.

- `\glsxtrpostlinkAddSymbolOnFirstUse` This will add the symbol in parentheses if that field is set and the hook follows the first use of the entry. This incorporates the inner formatting and symbol accessibility support, if provided.

- `\glsxtrpostlinkAddSymbolDescOnFirstUse` This will add the symbol, if that field is set, and the description (both within the same set of parentheses), if the hook follows the first use of the entry. This incorporates the inner formatting and accessibility support, if provided. The separator between the symbol and description is given by:

- `\glsxtrpostlinkSymbolDescSep` The default is a comma followed by a space.

For example:

```latex
\glsdefpostlink{general}{\glsxtrpostlinkAddDescOnFirstUse}
\glsdefpostlink{symbol}{\glsxtrpostlinkAddSymbolOnFirstUse}
\glsdefpostlink{number}{\glsxtrpostlinkAddSymbolDescOnFirstUse}
\newglossaryentry{sample}{name={sample},description={an example}}
\newglossaryentry{alpha}{name={alpha},symbol={\ensuremath{\alpha}},description={a symbol},category={symbol}}
\newglossaryentry{pi}{name={pi},symbol={\ensuremath{\pi}}},
```
5. Referencing (Using) Entries

\begin{document}
\description={a constant}, \category={number}

First use: \gls{sample}, \gls{alpha}, \gls{pi}.

Next use: \gls{sample}, \gls{alpha}, \gls{pi}.
\end{document}

This produces:

Example 103: Category post-link hooks

First use: sample (an example), alpha ($\alpha$), pi ($\pi$, a constant).
Next use: sample, alpha, pi.

The following commands are also provided for use in the post-link hook:

\glsxtrcurrentfield
\glsxtrifwasglslike\{⟨true⟩\}{⟨false⟩}
\glsxtrifwasglslikeandfirstuse\{⟨true⟩\}{⟨false⟩}
\glsxtrifwassubsequentuse\{⟨true⟩\}{⟨false⟩}

This expands to empty if the calling command isn’t associated with one specific field (such as \glslink, the \gls-like commands, the inline full form commands) otherwise it will expand to the name of the key associated with the singular form of the command. For example, this command will expand to text for both \gls{gls}{text} and \gls{gls}{plural}, to description for both \gls{gls}{desc} and \gls{gls}{descplural}, and to short for both \gls{gls}{short} and \gls{gls}{shortplural}. Whereas it will expand to nothing for both \gls{gls} and \gls{glsxtrfull}.

This expands to \langle true\rangle if the calling command was a \gls-like command and to \langle false\rangle otherwise.

This expands to \langle true\rangle if the calling command was a \gls-like command and was the first use otherwise it expands to \langle false\rangle. This is simply a shortcut command that uses both \glsxtrifwasglslike and \glsxtrifwasfirstuse.

This expands to \langle true\rangle if the calling command was a \gls-like command and was the subsequent
use otherwise it expands to \langle false\rangle. This is simply a shortcut command that uses both \glsxtrifwasglslike and \glsxtrifwasfirstuse.

\glsxtrifwassubsequentorshort{\langle true\rangle}{\langle false\rangle}

This expands to \langle true\rangle if the calling command was a \gls-like command and was the subsequent use or if the calling command set \glsxtrcurrentfield to short. Otherwise it expands to \langle false\rangle.

\glsxtrifallcaps{\langle all caps\rangle}{\langle not all caps\rangle}

This simply does:

\glscapscase{\langle not all caps\rangle}{\langle not all caps\rangle}{\langle all caps\rangle}

It’s not usually necessary for the post-link hook to differentiate between no case-change and sentence case, so this provides a convenient shortcut if only the all caps case needs to be different.

It’s possible you may also want to reference the inserted material. For the \gls-like commands, this can be obtained with the placeholder \glsinsert, but it’s not normally set by the \glstext-like commands, which don’t use the entry format style (§5.5.5) and instead incorporate the inserted material at the end of the link text. If you want the post-link hook to be able to access the inserted material for the \glstext-like commands, you must first save it, by redefining the following:

\glsxtrsaveinsert{\langle entry-label\rangle}{\langle insert\rangle}

This is used by the \glstext-like commands to initialise \glsinsert. The default is:

\newcommand*{\glsxtrsaveinsert}[2]{\def\glsinsert{}}

For example, to always save the insert:

\renewcommand*{\glsxtrsaveinsert}[2]{\def\glsinsert{#2}}

The first argument can be used to conditionally assign the insert. For example, the following will only save it for entries with the general category:
5. Referencing (Using) Entries

\renewcommand*{\glsxtrsaveinsert}[2]{% 
\glsifcategory{#1}{general}{\def\glsinsert{#2}}{\def\glsinsert{}}% }

If you only want to save the insert for the \glsxtrfull set of commands, you can redefine \glsxtrfullsaveinsert instead (see §4.3).

\glsxtrassignlinktextfmt

This contains the assignments required to ensure that \glslabel, \glstextformat and \glsxtrgenentrytextfmt have the definitions they had within the link text. They would ordinarily still have those definitions within the post-link hook, but if, for example, the hook contains content that may be deferred, such as a footnote, then judicious use and expansion of \glsxtrassignlinktextfmt can allow the deferred code to pick up the label, outer and inner formatting.

For example, the post-link hook might contain:

\expandafter\footnote\expandafter
{\glsxtrassignlinktextfmt\glstextformat{% \Glsaccessfmtdesc{}{\glsxtrgenentrytextfmt}{\glslabel}}.}

5.5.5. Entry Format Style

This section is for advanced users. Minor modifications can be made by adjusting the outer formatting (§5.5.1), the post-link hook (§5.5.4) or the abbreviation style commands (§4.5.1.3).

The \gls-like commands have the link text set to the entry format style corresponding to the entry’s glossary type. This can be changed with \defglsentryfmt, but the default style is given by \glsentryfmt, which is defined by the base glossaries package. This uses the placeholder commands to determine the appropriate text. These are described in the glossaries manual, but to recap they are: \glslabel (the entry’s label), \glscustomtext (text provided by \glsdisp or empty otherwise), \glsinsert (supplied in the final optional argument except for \glsdisp, empty by default), \glsifplural, and \glscapscase.

The glossaries-extra package redefines \glsentryfmt to test whether or not the entry is an abbreviation and, if so, whether or not the entry should be treated as a regular entry:
This uses \ifglsisshort to determine whether or not the entry is an abbreviation. If it is, then \glssetabbreviationfmt is used to setup the abbreviation style commands for the entry’s category.

Then there’s a check (with \glsisregular) to determine whether or not the entry should be treated as a regular entry. Note that if the regular attribute hasn’t been set to true, the entry will still be treated as a regular entry if it doesn’t have the short field set.

Regular entries are formatted according to:

\glsgenentryfmt

This is the generic regular entry format. It’s encapsulated with \glsxtrregularfont, but note that if the entry is an abbreviation, it will still use the abbreviation style formatting commands, which are contained within the first, firstplural, text and plural field values.

The generic regular entry format \glsgenentryfmt is provided by the base glossaries package, but is redefined by glossaries-extra to support inner formatting (§5.5.3) and accessibility (§9), if required.

Abbreviations that aren’t considered regular, are formatted according to:

\glsxtrgenabbreviationfmt

This is the generic non-regular abbreviation format. It’s encapsulated with \glsxtrabbreviationfont. Unlike \glsgenentryfmt this doesn’t reference the first, firstplural, text or plural fields, but instead uses the abbreviation formatting commands \glsxtrfullformat, \glsxtr subsequentfmt and their plural and case-changing variants.

If you want to define your own custom entry format, you will need to consider whether or not your format should support regular and non-regular abbreviation styles. Further detail can be found in the documented code:
5. Referencing (Using) Entries

5.6. Hyperlinks

The \gls-like and \glistext-like commands will automatically create a hyperlink by default, if hyperref has been loaded (before glossaries/glossaries-extra). The hyperlink can be switched off with hyper=false but will also be switched off if the entry was assigned to an ignored glossary that was defined with the unstarred \newignoredglossary.

The hyperfirst=false package option and the category attributes nohyper, nohyper-first and nohypernext can also be used to automatically switch off the hyperlink. See also the hyperoutside option that determines whether the hyperlink should be inside or outside of the outer formatting.

The hyperlink target is usually created by \glstarget which is used by all the predefined glossary styles by the standalone commands, such as \GlsXtrStandaloneEntryName. This can result in duplicate targets if you have multiple glossaries or both standalone entries and a glossary. There are ways of getting around this, such as changing the target prefix or using target=false when displaying the glossary. However, the simplest method is to redefine \glstarget to use:

\glsxstrtarget{⟨entry-label⟩}{⟨text⟩}

This behaves in a similar manner to \glstarget but first tests the field obtained by expanding:

\glsxstrtargetfield

By default, this expands to target. If this field is undefined (according to \GlsXtrIfField-Undef) the target will be created in the way that \glstarget would ordinarily create it (if hyperlinks are enabled). The field will then be set to the target. If the field has been defined then the target won’t be created and the ⟨text⟩ is simply displayed. In order to use this feature just redefine \glstarget:

\renewcommand{\glstarget}{\glsxstrtarget}

The target for an entry with the label ⟨entry-label⟩ is in the form ⟨prefix⟩⟨entry-label⟩. The ⟨prefix⟩ is normally \glolinkprefix but may be changed with the prefix option when displaying the glossary.
The target can also be changed to a link to an external file with the `targeturl` category attribute.

### 5.7. Label Prefixes

It’s possible that you may want to prefix labels to ensure uniqueness. For example, this manual references both the `\makeglossaries` command and the `makeglossaries` Perl script. They are both defined as glossary entries, but they can’t both have the label `makeglossaries`. This manual uses `bib2gls` and is quite complicated, but a simplified version is as follows:

```latex
\newcommand{\csfmt}[1]{\texttt{#1}}
\newcommand{\appfmt}[1]{\texttt{#1}}
\newglossaryentry{cs.makeglossaries}{name={\csfmt{makeglossaries}},
  description={}}
\newglossaryentry{app.makeglossaries}{name={\appfmt{makeglossaries}
  },
  description={}}
```

So the label `cs.makeglossaries` refers to `\makeglossaries` and `app.makeglossaries` refers to `makeglossaries`. If you have a lot of prefixes like this, you may prefer to have a command that automatically adds the prefix. For example,

```latex
\newcommand*{\cs}[2][\texttt]{{\gls[#1]{cs.#2}}}
```

The problem with this is that the custom command `\cs` doesn’t allow for the `*`, `+` and `⟨alt-mod⟩` modifiers (such as `\gls*` or `\gls+`). Instead you can use:

```latex\glsxtrnewgls[(default-options)]{⟨prefix⟩}{⟨cs⟩}
```

which defines the command

```latex
⟨cs⟩⟨modifier⟩[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]
```

that behaves like

```latex\gls⟨modifier⟩[⟨default options⟩,⟨options⟩]{⟨prefix⟩⟨entry-label⟩}[⟨insert⟩]
```

For example:
or (to default to no hyperlinks)
\glsxtrnewgls[hyper=false]{sym.}{\cs}

now you can use \cs+{M} to behave like \gls+{cs.M}.
If you also want the plural and sentence case versions you can use
\glsxtrnewglslike\langle default-options\rangle\langle prefix\rangle\langle\gls-like cs\rangle\langle\glspl-like cs\rangle\langle\Gls-like cs\rangle\langle\Glspl-like cs\rangle

For example:
\glsxtrnewglslike[hyper=false]{idx.}{\idx}{\idxpl}{\Idx}{\Idxpl}

For the all caps versions:
\glsxtrnewGLSlike\langle default-options\rangle\langle prefix\rangle\langle\GLS-like cs\rangle\langle\GLSpl-like cs\rangle

For example:
\glsxtrnewGLSlike[hyper=false]{idx.}{\IDX}{\IDXpl}

For commands that require the link text to be specified, you can use:
\glsxtrnewglslink\langle default-options\rangle\langle prefix\rangle\langle cs\rangle
which defines \langle cs\rangle\langle options\rangle\langle label\rangle\langle text\rangle to behave like \glslink\langle default-options\rangle\langle options\rangle\langle prefix\rangle\langle label\rangle\langle text\rangle, or
\glsxtrnewglsdisp\langle default-options\rangle\langle prefix\rangle\langle cs\rangle
which defines \langle cs\rangle\langle options\rangle\langle label\rangle\langle text\rangle to behave like \glsdisp\langle default-options\rangle\langle options\rangle\langle prefix\rangle\langle label\rangle\langle text\rangle.
5. Referencing (Using) Entries

If you are using bib2gls, it can pick up the custom commands that are defined using the above, so it can detect dependencies when it parses fields such as description. If you provide your own custom command with just \newcommand that has syntax that starts with \[⟨options⟩⟩{⟨entry-label⟩}, then you can notify bib2gls using:

\glsxtridentifyglslike{⟨label-prefix⟩}{⟨cs⟩}

where ⟨label-prefix⟩ is the prefix to apply to the label that’s passed to the command ⟨cs⟩. The information is written to the aux file so that bib2gls can add the given command to those it looks for when searching for dependencies.

Another possibility when using bib2gls is to set up known label prefixes, see §11.5.7 for further details.

If you use bib2gls with record counting, there are commands to \glsxtrnewgls for \rgls:

\glsxtrnewrgls[⟨default-options⟩]{⟨prefix⟩}{⟨cs⟩}

and for \rgls, \rglsp, \rGls and \rGlspl:

\glsxtrnewglslike[⟨default-options⟩]{⟨prefix⟩}{⟨\rgls-like cs⟩}{⟨\rglsp-like cs⟩}{⟨\rGls-like cs⟩}{⟨\rGlspl-like cs⟩}

and for all caps:

\glsxtrnewrGLSlke[⟨default-options⟩]{⟨prefix⟩}{⟨rGLS-like cs⟩}{⟨rGLSpl-like cs⟩}

Defining commands in this manner (rather than simply using \newcommand) also allows the command to be identified as a sentence case blocker to prevent the label from being converted or, in the case of \glsxtrnewglslike and \glsxtrnewrglslike, as a mapping. See §5.2 for further details.

5.8. Indexing

Indexing is normally performed implicitly by the \gls-like and \glstext-like commands, but this action can be prevented, such as by using the option noindex=true. These commands also generate text (the link text, §5.5). If you want to simply index an entry (to ensure that an entry is shown in the glossary) without producing any text then you can use \glsadd. Indexing is also performed by cross-referencing commands, such as \glssee. In the case of
makeindex, \glssee simply behaves like \glsadd with a special format and the location set to Z (which pushes it to the end of the location list). Entries in ignored glossaries can only be indexed with \bib2gls.

If you want all defined entries to appear in the glossary, regardless of whether or not they have been used in the document, then you can use \glsaddall or \glsaddallunused (both provided by the base glossaries package). These both iterate over all entries (in all non-ignored glossaries). In the first case (\glsaddall), every entry is indexed with the \glsadd options provided in the optional argument of \glsaddall. In the second case (\glsaddallunused), only those entries that haven’t been marked as used so far will be indexed using \glsadd[format=glsignore]{⟨label⟩}. See the glossaries manual for further details of those commands.

The glossaries-extra package provides a similar command:

\glsaddallunindexed[(glossary types)]

This is like \glsaddallunused but indexes all entries that haven’t been indexed so far (again using the option format=glsignore). This is preferable to \glsaddallunused if you have to reset the first use flag for any entries. As with \glsaddallunused, if this command is required, it should be placed near the end of the document. Indexing any entries after either of these commands are used will cause spurious commas in the location lists.

Iterative commands such as \glsaddall, \glsaddallunused and \glsaddallunindexed should not be used with \bib2gls. Use the selection=all option instead.

If you want to index a specific subset of entries, rather than all entries for a given glossary, you can use:

\glsaddeach[(options)]{(entry label list)}

This does \glsadd[(options)]{(entry-label)} for each entry in the comma-separated ⟨entry label list⟩. This command may be used with \bib2gls, although it may be simpler to adjust the selection criteria or use filtering.

Explicit ranges can be formed by including ⟨(range start) and ⟨range end) at the start of the format value. For example:

\glsadd[format=⟨]{example}
...
\glsadd[format=⟨]{example}
(See the glossaries manual for further details.) However, the isolated open and close parentheses can upset syntax highlighting. So the glossaries-extra package provides the following commands, which automatically add ( and ).

\texttt{\textbackslash glsstartrange\[\langle options\]\{\langle entry label list\}\]}

This effectively does:

\texttt{\textbackslash glsaddeach\[\langle options\>,format=\langle\texttt{encap}\]\{\langle entry-label list\}\]}

\texttt{\textbackslash glsendrange\[\langle options\]\{\langle entry label list\}\]}

This effectively does:

\texttt{\textbackslash glsaddeach\[\langle options\>,format=\texttt{\langle\texttt{encap}\}\]\{\langle entry-label list\}\]}

The default value of \langle\texttt{encap}\rangle will be the same as the default number format (which can be changed with \texttt{\textbackslash GlsXtrSetDefaultNumberFormat}). If you want a different default for ranges, use:

\texttt{\textbackslash GlsXtrSetDefaultRangeFormat\{\langle\texttt{encap}\}\]}

This sets the default format for \texttt{\textbackslash glsstartrange} and \texttt{\textbackslash glsendrange}. Note that this format won't be applied if you explicitly create a range with \texttt{\textbackslash glsadd} or \texttt{\textbackslash glsaddeach}.

Alternatively, you can use \texttt{format=\texttt{encap}} in \langle\texttt{options}\rangle, but remember that this will need to be the same in both \texttt{\textbackslash glsstartrange} and \texttt{\textbackslash glsendrange}. For example:

\texttt{\textbackslash glsstartrange[format=\texttt{hyperbf}]\{example\} \dots \textbackslash glsendrange[format=\texttt{hyperbf}]\{example\}}

This is the same as:

\texttt{\textbackslash GlsXtrSetDefaultRangeFormat\{\texttt{\textbackslash hyperbf}\} \textbackslash glsstartrange\{example\}
5. Referencing (Using) Entries

```latex
\glsendrange{example}
```

which is the same as:

```latex
\glsadd[format=(hyperbf)]{example}
```

The mandatory argument of `\glsstartrange` and `\glsendrange` may be a comma-separated list of entry labels. For example:

```latex
\glsstartrange{duck,goose}
```

This is essentially the same as:

```latex
\glsadd[format=]{duck}\
\glsadd[format=]{goose}
```

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

This will make the `\gls`-like and `\glstext`-like commands automatically use `\glsadd{⟨glsadd options⟩}{⟨entry-label⟩}` whenever a `\gls`-like or `\glstext`-like command is used for the entry given by `⟨entry-label⟩` when the `format` matches one of the formats in the comma-separated `⟨format list⟩`.

The optional argument `⟨label⟩` defaults to `⟨entry-label⟩` (which will match `⟨entry-label⟩` that was used with `\gls` etc) and indicates the entry label to use in `\glsadd` and so needs to be expandable. The `⟨format list⟩` is a comma-separated list of format values that will trigger the automated adding. The `⟨glsadd options⟩` are the options to pass to `\glsadd` with `format=⟨format⟩` prepended to the list.

For example, with:
5. Referencing (Using) Entries

\GlsXtrAutoAddOnFormat{hyperbf}{counter=chapter}

then \gls{format=hyperbf}{sample} will be equivalent to:

\glsadd{format=hyperbf,counter=chapter}{sample}\gls{format=hyperbf}{sample}

Note that the explicit range markers will prevent a match unless you include them in \textit{(format list)} (in which case, be sure to add both the start and end formats).

Here’s another example:

\GlsXtrAutoAddOnFormat[dual.\glslabel]{hyperbf}{}

In this case \gls{format=hyperbf}{sample} will now be equivalent to:

\glsadd{format=hyperbf}{dual.sample}\gls{format=hyperbf}{sample}

\GlsXtrAutoAddOnFormat is not applied to \glsadd as it could cause an infinite loop.

In the context of glossaries and glossaries-extra, indexing refers to the mechanism used to ensure that an entry is included in its associated glossary. (If you also want to use \index, see §12.) This includes any entries that simply cross-reference another entry. The default is to use makeindex, which is a general purpose indexing application. Each time an entry is indexed, a line is added to an associated file that contains the indexing information, which includes the sort value, the hierarchical information (if the entry has a parent) and an associated location (the page number, by default). This information is used to sort the entries and collate the locations into a compact location list. The \xindy package option switches to using \xindy syntax, but the process is much the same.

Since both makeindex and xindy are general purpose indexing applications they require an associated location (or a cross-reference) since indexes are typically used to lookup the locations in the document where the term occurs. Although glossaries are similar to indexes they can simply be used to provide brief summaries of each term without any locations. The way that makeindex and xindy work means that valid locations (that is, locations that conform to makeindex/xindy syntax) must be supplied even if no location list is required. If an invalid location is used, an error will occur during the makeindex/xindy step in the build process, even if the location will eventually be ignored when typesetting the glossary.

All location lists can be suppressed with the \nonumberlist option (which simply discards the location list for each entry), but there are occasions where only some locations need to be
5. Referencing (Using) Entries

suppressed. The main way of hiding a location is to encapsulate the location with a command that does nothing. The \glsignore command is used for this purpose (format=glsignore). However, it’s important to remember that even though the location isn’t shown, it’s still present in the location list. This means that you will end up with spurious commas if there’s more than one item in the location list.

The “noidx” method similarly writes indexing information, but in this case the information is written to the aux file. Again, empty locations can cause spurious commas in the location lists.

The only method that recognises \glsignore as a special “ignored location” is bib2gls, where this format will trigger the entry’s selection but won’t add the ignored location to the location list. This avoids the problem of spurious commas caused by invisible locations.

The location corresponds to a counter. The default is the page counter, but may be changed with the counter package option, the ⟨counter⟩ optional argument of \newglossary, the counter key when defining an entry, or the counter option when indexing an entry.

Note that bib2gls v3.0+ converts an empty location (which can occur when the location counter is 0 and should be formatted as a Roman numeral) to an ignored location. For example, if you use counter=part but have \gls before the first \part. An empty location will trigger an error with makeindex and xindy.

Since no entries are defined on the first \LaTeX run with bib2gls, there’s no way of determining the entry’s glossary type or of finding if the entry’s counter key has been set. This means that if the counter has been assigned to either the entry’s glossary or to the entry itself, the location counter can’t be implemented until the entry has been defined. A second build is required to ensure that the locations use the correct counter.

The location counter must expand to syntax that’s recognised by the indexing application. This is very restrictive with makeindex, which only recognises Western Arabic (\arabic), lowercase Roman numerals (\roman), uppercase Roman numerals (\Roman), lowercase Basic Latin (\alph) and uppercase Basic Latin (\Alph), with optionally a separator (hyphen by default). With xindy, the syntax must be defined (see the glossaries manual for further details).

There’s no restriction on the location syntax with bib2gls. The only limitation is that if bib2gls can’t determine an associated numeric value according to its location parser, it won’t form ranges. This means that with bib2gls, you can set arbitrary text as the location (that’s not related to a counter) with thevalue. You can also use thevalue with makeindex and xindy, but only if the value matches the required location syntax.

Both makeindex and xindy order the locations in the location lists. For example:

```
\makeglossaries
\newglossaryentry{sample}{name={sample},description={an example}}
\begin{document}
```
5. Referencing (Using) Entries

\[\text{\begin{filecontents*}{\jobname.bib} @entry{sample, name={sample}, description={an example}} \end{filecontents*}} \]
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources
\begin{document}
\gls{\text{\thevalue=Z}}{\text{sample}} \text{(Z), } \gls{\text{\thevalue=4}}{\text{sample}} \text{(4), }
\gls{\text{\thevalue=xi}}{\text{sample}} \text{(xi), } \gls{\text{\thevalue=2}}{\text{sample}} \text{(2), }
\gls{\text{\thevalue=iii}}{\text{sample}} \text{(iii), } \gls{\text{\thevalue=A}}{\text{sample}} \text{(A).}
\printglossaries
\end{document} \]

Example 104: Location list ordering (\texttt{makeindex})

\begin{verbatim}
sample (Z), sample (4), sample (xi), sample (2), sample (iii), sample (A).
\end{verbatim}

Glossary

\begin{verbatim}
sample an example iii, xi, 2, 4, A, Z
\end{verbatim}

With \texttt{makeindex}, the location list is grouped into the different number formats (\texttt{\roman}, \texttt{\arabic} and \texttt{\Alph}), with each group ordered numerically. The same result can be produced with \texttt{xindy} by adding the \texttt{xindy} package option to the above example.

With \texttt{bib2gls}, the location list is always in order of indexing. The above example document can be converted to use \texttt{bib2gls} as follows:

\begin{verbatim}
\begin{filecontents*}{\jobname.bib} @entry{sample, name={sample}, description={an example}} \end{filecontents*}
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources
\begin{document}
\gls{\text{\thevalue=Z}}{\text{sample}} \text{(Z), } \gls{\text{\thevalue=4}}{\text{sample}} \text{(4), }
\gls{\text{\thevalue=xi}}{\text{sample}} \text{(xi), } \gls{\text{\thevalue=2}}{\text{sample}} \text{(2), }
\gls{\text{\thevalue=iii}}{\text{sample}} \text{(iii), } \gls{\text{\thevalue=A}}{\text{sample}} \text{(A).}
\printunsrtglossaries
\end{document}
\end{verbatim}
5. Referencing (Using) Entries

Example 105: Location list ordering (bib2gls)

sample (Z), sample (4), sample (xi), sample (2), sample (iii), sample (A).

Glossary

sample an example Z, 4, xi, 2, iii, A

This example is contrived. For most documents, the order of indexing will likely match the desired location list order.

Another important difference between bib2gls and the other indexing methods is the treatment of cross-references identified by the cross-reference keys see, seealso and alias. With bib2gls, the cross-referencing information is picked up when bib2gls parses the bib file and is used to establish dependencies, which ensures that when entries with cross-references are selected, their cross-referenced entries will also be selected.

With the other methods, cross-references are added to an entry’s location list by indexing the entry with a special format. The see, seealso and alias keys automatically trigger this indexing unless autoseeindex=false. See §5.9 for further details.

Every time an entry is indexed, the following hook is also used:

\glsxtrdowrglossaryhook{⟨entry-label⟩}

This does nothing by default. The argument is the entry’s label.

The indexing code is encapsulated with:

\glsencapwrcontent{⟨code⟩}

This adds grouping, which helps to prevent spacing issues caused by the whatsis that’s creates by the indexing.

The base glossaries package always performs the indexing before the link text for the \gls- like and \glstext-like commands. This means that if a page break occurs in the middle of the link text, the location will refer to the page number at the start of the link text (assuming the default page location counter). With glossaries-extra, you can use the option wrgloss= after to have the indexing occur after the link text. The wrgloss attribute can also be used. The default setting is initialised with \glsxtrinitwrgloss (see §5.1.1).

Every time an entry is indexed, an internal field associated with the entry’s label is globally updated to keep a count of the number of times the entry has been indexed. The value can
be accessed with:

```
\glsentryindexcount{⟨entry-label⟩}
```

This command will expand to 0 if the entry hasn’t been indexed or hasn’t been defined. To test if the value is greater than 0 (that is, to test if the entry has been indexed yet), use:

```
\glsifindexed{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}
```

This expands to ⟨true⟩ if the entry is defined and has been indexed, otherwise it expands to ⟨false⟩. No warning or error occurs if the entry hasn’t been defined.

Note that the index count is a running total. This is not the same as the record count saved by bib2gls’s --record-count switch, which represents the total number of records for the given entry from the previous \LaTeX run.

The base glossaries package defines:

```
\glswriteentry{⟨entry-label⟩}{⟨code⟩}
```

This command conditionally writes the indexing code (supplied by the second argument ⟨code⟩). The original definition simply tests whether or not the indexonlyfirst setting is on. The glossaries-extra package redefines this command to perform additional checks to determine whether or not the indexing code should be performed.

The modified definition uses:

```
\glsxtrifindexing{⟨true⟩}{⟨false⟩}
```

to test the noindex setting. This does ⟨false⟩ if noindex=true, otherwise it does ⟨true⟩.

```
\ifglsindexonlyfirst ⟨true⟩\else ⟨false⟩\fi
```

This is a conditional that corresponds to the indexonlyfirst package option. First use is tested using \GlsXtrIfUnusedOrUndefined rather than \ifglsused. The indexonlyfirst attribute is also tested. If the “index only first” setting is on and the entry has been used, ⟨code⟩ isn’t performed but auto-indexing via \glsxtrdoautoindexname is still performed (see §12).
5. Referencing (Using) Entries

Since no entries are defined on the first \LaTeX run with \texttt{bib2gls}, there’s no way of keeping track of whether or not an entry has been used or what its \texttt{category} is, which is required to query the \texttt{indexonlyfirst} attribute, so for the first document build all instances will be indexed. A second build is required for the “index only first” feature.

5.9. Cross-Referencing

The base \texttt{glossaries} package only provides the \texttt{see} key, which automatically indexes the cross-reference using \texttt{\glssee}. The value of this key isn’t saved and can’t be accessed later. (The key was simply provided as a shortcut.) The indexing ensures that the cross-reference is shown in the location list.

The auto-indexing feature of the \texttt{see} key was intended as a shortcut where only entries required in the document are defined. If you want to have a large file containing entries that may or may not be required in the document, then using \texttt{see}, \texttt{seealso} or \texttt{alias} can cause unwanted entries to appear in the glossary. In this case, see §5.9.1.

The \texttt{glossaries-extra} package saves the value of the \texttt{see} key and additionally provides the \texttt{seealso} and \texttt{alias} keys that perform similar functions. The values of the \texttt{see}, \texttt{seealso} and \texttt{alias} keys can all be accessed at a later point in the document.

If an entry with a cross-reference has been included in the glossary, there’s no guarantee that the cross-referenced entry will also be included. It won’t be included if it hasn’t been indexed anywhere in the document. You can use the \texttt{indexcrossrefs} package option to search for cross-references that require indexing at the end of the document, but note that this can be time-consuming if you have a large number of entries.

With \texttt{bib2gls} you can simply change the selection criteria (\texttt{selection=\{recorded and deps and see\}} or \texttt{selection=\{recorded and deps and see not also\}}) to ensure that all cross-referenced entries are included even if they haven’t been indexed in the document.

Example (\texttt{see} and \texttt{seealso} keys):

\begin{verbatim}
\newglossaryentry{pumpkin}{name={pumpkin},description={}}
\newglossaryentry{cucumber}{name={cucumber},description={}}
\newglossaryentry{melon}{name={melon},description={}}
\newglossaryentry{gourd}{name={gourd},description={}}
\end{verbatim}
5. Referencing (Using) Entries

```
\newglossaryentry{gourd}{name={gourd},description={},
see={\[related topics\]pumpkin,cucumber,melon}}
```

When the gourd entry is defined, the cross-reference will automatically be indexed using \glssee. This means that the gourd entry will appear in the glossary, regardless of whether or not it is used in the document, with “see pumpkin, cucumber & melon” in the location list. If gourd is also indexed in the document, then those locations will also be added to the gourd’s location list.

The cross-referenced entries (pumpkin, cucumber and melon) will only appear in the glossary if they are also indexed in the document. This can be implemented automatically with indexcrossrefs.

The seealso key in the marrow entry functions in much the same way, but it is indexed with \glsttrindexseealso. This means that the marrow entry will have “see also courgette” in its location list.

The see key may optionally start with \langle\{tag\}\rangle to replace the default \seename tag with \\langle\{tag\}\rangle. The seealso key doesn’t permit this. For example, the following is permitted:

```
\newglossaryentry{gourd}{name={gourd},description={},
see={\[\{related topics\}\]pumpkin,cucumber,melon}}
```

but you can’t replace see with seealso in the above as it would assume that the first label in the list is \{related topics\}pumpkin which is incorrect. The tag would have to be removed:

```
\newglossaryentry{gourd}{name={gourd},description={},
seealso={pumpkin,cucumber,melon}}
```

(You could then redefine \seename to \{related topics\}, if required or redefine \glsttruseseealsoformat as applicable.)

Example (alias key):

```
\newglossaryentry{zucchini}{name={zucchini},description={},
alias={courgette}}
```

When the zucchini entry is defined, the alias key will automatically index zucchini with \glssee{zucchini}{courgette}. This means that the zucchini entry will be present in the glossary with “see courgette” in the location list. If the zucchini entry is referenced in the document using a command like \glx, then the hyperlink (if enabled) will go to the courgette entry (not the zucchini entry) but the zucchini entry won’t be indexed.
5. Referencing (Using) Entries

If you want the zucchini entry locations added to the courgette entry, you can redefine `\glsetaliasnoindex` (see §5.9.3) or, with `bib2gls`, use the `alias-loc=transfer` setting.

With `bib2gls`, cross-references are selected according to the selection criteria. See the `bib2gls` manual for further details.

5.9.1. Entries that may not be required

If you have a file containing a large number of entry definitions shared across multiple documents, then the use of the `see`, `seealso` or `alias` key can cause unwanted entries to appear in the document. This can be demonstrated as follows. Suppose the file `myentries.tex` contains:

```latex
\newglossaryentry{pumpkin}{name={pumpkin},description={}}
\newglossaryentry{cucumber}{name={cucumber},description={}}
\newglossaryentry{melon}{name={melon},description={}}
\newglossaryentry{gourd}{name={gourd},description={}, see={pumpkin,cucumber,melon}}
\newglossaryentry{cucurbit}{name={cucurbit},description={}, see={gourd}}
\newglossaryentry{courgette}{name={courgette},description={}}
\newglossaryentry{marrow}{name={marrow},description={}, seealso={courgette}}
\newglossaryentry{zucchini}{name={zucchini},description={}, alias={courgette}}
\newglossaryentry{broccoli}{name={broccoli},description={}}
\newglossaryentry{cauliflower}{name={cauliflower},description={}, seealso={broccoli}}
```

Some of these entries have a cross-reference key set, but not all of these entries are required in the document:

```latex
\usepackage[colorlinks]{hyperref}
\usepackage[nostyles,stylemods=bookindex,style=bookindex] {glossaries-extra}
\makeglossaries
\loadglsentries{myentries}
\begin{document}
```

271
This document is only discussing \textit{courgette} (baby \textit{marrow}, also called a \textit{zucchini}), \textit{pumpkin} and \textit{melon}.

\printglossaries
\end{document}

Example 106: Cross-references (autoseeindex=true)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

\begin{center}
\textbf{Glossary}
\end{center}

\begin{tabular}{ll}
\textbf{C} & \textbf{M} \\
cauliflower & marrow 1, see also courgette \\
\textit{courgette} 1 & melon 1 \\
cucurbit & pumpkin 1 \\
\textit{gourd} & \textit{zucchini} see courgette \\
\end{tabular}

Note that the glossary includes cucurbit and gourd, which aren’t referenced in the document. They could be useful as a redirect for the reader, but the gourd entry cross-references the cucumber entry, which isn’t included in the glossary, so the hyperlink target is undefined. The cauliflower entry has also been included in the glossary, but in this case it’s not useful for the reader as neither cauliflower nor broccoli (which it cross-references) are mentioned in the document. As with the cucumber cross-reference, the broccoli cross-reference hyperlink target is undefined.

There are a number of methods to address some of these problems. The first method has the cross-referencing keys in the \texttt{tex} file (as above), but disables the auto-indexing:

\begin{verbatim}
\usepackage[colorlinks]{hyperref}
\usepackage[autoseeindex=false,nostyles,stylemods=bookindex, style=bookindex]{glossaries-extra}
\makeglossaries
\loadglsentries{myentries}
\end{verbatim}
5. Referencing (Using) Entries

This document is only discussing \textit{courgettes} (baby \textit{marrows}, also called a \textit{zucchini}), \textit{pumpkins} and \textit{melons}.

\textbf{Glossary}

\begin{tabular}{ll}
\textit{C} & melon 1 \\
\textit{courgette} 1 & \\
\textit{M} & \\
\textit{marrow} 1 & pumpkin 1 \\
\textit{P} & \\
\end{tabular}

This doesn’t show the zucchini entry or any of the cross-references in the glossary because the information hasn’t been added to the indexing files. One way around this is to insert the cross-reference in a post-description hook.
This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

<table>
<thead>
<tr>
<th>C</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>courgette</td>
<td>pumpkin</td>
</tr>
<tr>
<td>marrow, see also</td>
<td>courgette</td>
</tr>
</tbody>
</table>

However, this still doesn’t solve the problem that the zucchini entry isn’t included in the glossary. It needs to be indexed, but indexing has been suppressed. Firstly, because the automatic indexing triggered by the alias key has been suppressed with autoseeindex=false, and, secondly, because the presence of the alias key automatically suppresses indexing with the \gls-like and \glstext-like commands. This doesn’t cause a problem for the zucchini hyperlink, since the target is courgette (obtained from the alias key).

The second method is to not use those keys in the entry definitions and use \glssee or \glsextrindexseealso within the document. For example, the file myentries.tex now contains:

\newglossaryentry{pumpkin}{name={pumpkin},description={}}
\newglossaryentry{cucumber}{name={cucumber},description={}}
\newglossaryentry{melon}{name={melon},description={}}
\newglossaryentry{gourd}{name={gourd},description={}}
\newglossaryentry{cucurbit}{name={cucurbit},description={}}
\newglossaryentry{courgette}{name={courgette},description={}}
\newglossaryentry{marrow}{name={marrow},description={}}
\newglossaryentry{zucchini}{name={zucchini},description={}}
\newglossaryentry{broccoli}{name={broccoli},description={}}
\newglossaryentry{cauliflower}{name={cauliflower},description={}}

The document:

\usepackage[colorlinks]{hyperref}
\usepackage[nostyles,stylemods=bookindex,style=bookindex]
5. Referencing (Using) Entries

```latex
{glossaries-extra}
\makeglossaries
\loadglsentries{myentries}
\glssee{gourd}{pumpkin, melon, courgette}
\glssee{zucchini}{courgette}
\GlsXtrSetField{zucchini}{alias}{courgette}
\begin{document}
This document is only discussing \gls{courgette} (baby\glsp{marrow}, also called a \gls{zucchini}), \gls{pumpkin} and \gls{melon}.
\printglossaries
\end{document}
```

Example 109: Cross-references (no see,seealso or alias)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

<table>
<thead>
<tr>
<th>C</th>
<th>melon 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>courgette 1</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>gourd</td>
<td>see pumpkin, melon &amp; courgette</td>
</tr>
<tr>
<td>marrow 1</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>pumpkin 1</td>
</tr>
<tr>
<td>Z</td>
<td></td>
</tr>
</tbody>
</table>

Note that aliases require the alias field to be set. In this case, I’ve set it with \GlsXtrSetField. The gourd and zucchini entries have been included in the glossary because they were added with \glssee. The other entries are in the glossary because they were indexed when referenced with \gls or \glsp.

Since cucumber isn’t required in the document, I haven’t included it in the cross-reference list for gourd. This method is flexible as it allows the cross-referencing to vary between documents. For example, another document may instead have:
5. Referencing (Using) Entries

The third method is to switch to bib2gls. The file myentries.tex can be converted to myentries.bib using:

```
convertgls2bib --index-conversion myentries.tex myentries.bib
```

I’ve used the option --index-conversion (or -i) which will use @index instead of @entry for entries that have an empty description (which is the case in this example). This creates the file myentries.bib, which contains the following (space compacted):

```latex
% Encoding: UTF-8
@index{pumpkin, name={pumpkin}}
@index{cucumber, name={cucumber}}
@index{melon, name={melon}}
@index{gourd, see={pumpkin,cucumber,melon}, name={gourd}}
@index{cucurbit, see={gourd}, name={cucurbit}}
@index{courgette, name={courgette}}
@index{marrow, name={marrow}, seealso={courgette}}
@index{zucchini, name={zucchini}, alias={courgette}}
@index{broccoli, name={broccoli}}
@index{cauliflower, name={cauliflower}, seealso={broccoli}}
```

The earlier example 106 on page 272 can be rewritten as:

```latex
\usepackage[colorlinks]{hyperref}
\usepackage[record,nostyles,stylemods=bookindex,style=bookindex]{glossaries-extra}
\GlsXtrLoadResources[src=myentries]
\begin{document}
This document is only discussing \gls{courgette} (baby \gls{marrow}, also called a \gls{zucchini}), \gls{pumpkin} and \gls{melon}.
\printunsrtglossaries
\end{document}
```

In order to support letter groups, bib2gls needs to be invoked with the --group switch. The result is:

276
5. Referencing (Using) Entries

Example 110: Cross-references (bib2gls)

This document is only discussing courgettes (baby marrows, also called a zucchini), pumpkins and melons.

Glossary

<table>
<thead>
<tr>
<th>C</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>courgette, 1</td>
<td>pumpkin, 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>marrow, 1, see also courgette</td>
<td>zucchini, see courgette</td>
</tr>
</tbody>
</table>

This uses the default selection={recorded and deps}, which selects entries that have records, and their dependencies. Records correspond to the usual indexing performed by the \gls-like, \glstext-like or \glsadd commands. With bib2gls, the cross-referencing fields don’t trigger an index but identify dependencies.

Note that the above doesn’t include the gourd entry (which cross-references entries that have been indexed). The selection criteria can be changed to also include unrecorded entries that cross-reference selected entries. There are two options to choose from: selection={recorded and deps and see}, which will apply to all cross-reference fields (see, see-also and alias), or selection={recorded and deps and see not also}, which doesn’t consider the seealso field.

Changing the resource options in the above example to:

\GlsXtrLoadResources[src=myentries, selection={recorded and deps and see}]

results in:
5. Referencing (Using) Entries

Example 111: Cross-references (\texttt{bib2gls} and \texttt{selection=recorded and deps and see})

This document is only discussing \textit{courgettes} (baby marrows, also called a zucchini), pumpkins and melons.

\begin{center}
\begin{tabular}{ll}
\textbf{C} & \textbf{M} \\
\textit{courgette}, 1 & marrow, 1, \textit{see also courgette} \\
cucumber & melon, 1 \\
cucurbit, \textit{see gourd} & \\
\textbf{G} & \textbf{P} \\
gourd, \textit{see pumpkin, cucumber \& melon} & pumpkin, 1 \\
\textbf{Z} & \\
zucchini, \textit{see courgette} &
\end{tabular}
\end{center}

This now includes the gourd entry because it cross-references pumpkin and melon, which have been recorded in the document. The cucurbit entry is also included because it cross-references the (now selected) gourd entry. Note that the cucumber entry has been selected because the gourd entry depends on it. This means there are no broken links in the glossary, but it looks a bit odd as the cucumber entry has no location list. As from \texttt{bib2gls v3.0}, this can be removed with one of the cross-reference pruning options, such as \texttt{prune-xr}. For example:

\texttt{\GlsXtrLoadResources[src=myentries, selection={recorded and deps and see},prune-xr]}

results in:

\texttt{278}
Example 112: Cross-references (\texttt{bib2gls} and \texttt{selection=recorded} and \texttt{deps} and \texttt{see,prune=xr})

This document is only discussing \textit{courgettes} (baby marrows, also called a zucchini), pumpkins and melons.

\begin{center}
\begin{tabular}{ll}
\textbf{C} & melon, 1 \\
courgette, 1 & \\
cucurbit, \textit{see gourd} & \\
\textbf{G} & pumpkin, 1 \\
gourd, \textit{see pumpkin \& melon} & \\
\textbf{M} & \\
marrow, 1, \textit{see also courgette} & \\
\textbf{Z} & \\
zucchini, \textit{see courgette} & \\
\end{tabular}
\end{center}

This has removed the unnecessary cucumber from the gourd's \textit{see list}, and so cucumber doesn't get selected.

See the \texttt{bib2gls} user manual for further details on the cross-reference selection and pruning options.

\subsection*{5.9.2. Accessing the Cross-Referencing Fields}

If you have switched off the indexing of the cross-reference fields (with \texttt{autoseeindex=\texttt{false}}) or want to suppress the location lists, then you can adjust the glossary style or hooks to include the cross-references since they won't be shown otherwise.

\begin{itemize}
\item \texttt{\textlsxtrseelists\{\langle entry-label\rangle\}}:
\end{itemize}

If the entry given by \langle\textit{entry-label}\rangle has the \textit{see}, \textit{seealso} or \textit{alias} fields set, this will display the cross reference according to \texttt{\textlsxtruseseeformat} (for \textit{see} and \textit{alias}) or \texttt{\textlsxtruseseealsoformat} (for \textit{seealso}). If any of these fields are set, the list is encapsulated with:

\begin{itemize}
\item \texttt{\textlsxtrseelistsencap\{\langle content\rangle\}}:
\end{itemize}
5. Referencing (Using) Entries

This simply does a space followed by \textlangle\textit{content}\traget. If more than one of the fields are set (not recommended), then they will be displayed in the order: see, seealso and alias. The entire set will be encapsulated with \texttt{\glsvalxtrseelistsencap} and each sub-list will be separated with:

\texttt{\glsvalxtrseelistsdelim}

which defaults to a comma followed by a space.

\texttt{\glsvalxtrusesee\{\textit{entry-label}\}}

If the entry given by \textit{entry-label} has the see field set, this will display the cross reference according to \texttt{\glsvalxtruseseeformat}, otherwise this does nothing. An error (or warning with \texttt{undefaction=warn}) will occur if the entry hasn’t been defined.

\texttt{\glsvalxtrusealias\{\textit{entry-label}\}}

As \texttt{\glsvalxtrusesee} but for the alias field.

\texttt{\glsvalxtruseseealso\{\textit{entry-label}\}}

If the entry given by \textit{entry-label} has the seealso field set, this will display the cross reference according to \texttt{\glsvalxtrusealsoformat}, otherwise this does nothing. An error (or warning with \texttt{undefaction=warn}) will occur if the entry hasn’t been defined.

\texttt{\glsvalxtrialias\{\textit{entry-label}\}}

This expands to the value of the alias field (which should be a single entry label) or empty if the field isn’t set. If the entry isn’t defined, this command will expand to \texttt{\relax} (without any error or warning). If you want to first test if the field is set, you can use \texttt{\glsvalxtrif\text{-}field}.

\texttt{\glsvalxtrusealsolabels\{\textit{entry-label}\}}

This expands to the value of the seealso field (which should be a comma-separated list of entry labels) or empty if the field isn’t set. If the entry isn’t defined, this command will expand to \texttt{\relax} (without any error or warning). If you want to first test if the field is set,
you can use \glsxtrusesealsoformat.

\glsxtrusesealsoformat{\csvlist}

This command is used to format a “see also” cross-reference. This is simply defined to do:

\glsseeformat[\seealso\name]{\csvlist}{()}\]

**5.9.3. Cross-Reference Indexing**

If you are using bib2gls, see the bib2gls user manual for information about the selection, see, seealso, alias, alias-loc options.

The actual indexing of the seealso key is performed with:

\glsxtrindexseealso{\entrylabel}{\xrlist}

which is analogous to \glssee. As with \glssee, this can also be used explicitly.

With makeindex, \glsxtrindexseealso simply does:

\glssee[\seealso\name]{\entrylabel}{\xrlist}

With xindy, \glsxtrindexseealso behaves in an analogous way, using the appropriate cross-referencing markup.

\glsxtrsetaliasnoindex

This hook is used within the \gls-like and \glistext-like commands to automatically switch off the indexing for aliases. (The hook is performed after the options set by \GlsXtrSet-DefaultGlsOpts.)

By default, this hook just sets noindex=true. If you would like to add locations to the aliased location list then you can redefine it to use:

\glsxtrindexaliased

For example:
5. Referencing (Using) Entries

\renewcommand{\glsxtrsetaliasnoindex}{\glsxtrindexaliased}

Note that this needs \texttt{noindex=false} to ensure the indexing takes place so don’t simply append \texttt{\glsxtrindexaliased} to the definition of \texttt{\glsxtrsetaliasnoindex}.

Don’t use the above hooks with \texttt{bib2gls} as this function is disabled with \texttt{record=only} and \texttt{record=nameref}. Use the \texttt{alias-loc} resource option instead.

\glsxtraddallcrossrefs

This is used at the end of the document if \texttt{indexcrossrefs=true} to automatically index any cross-references (identified in the \texttt{see}, \texttt{seealso} and \texttt{alias} fields). This command iterates over all entries in all glossaries and, if an entry has been marked as used, does:

\glsxtraddunusedxrefs

which indexes any labels identified in the cross-reference fields of the entry given by \texttt{⟨entry-label⟩} that haven’t been marked as used.

This can be time consuming if there are a large number of entries defined. If this is the case, you may want to consider switching to \texttt{bib2gls} and use either \texttt{selection={recorded and deps and see}} or \texttt{selection={recorded and deps and see not also}}.

There should be no need to use \texttt{\glsxtraddallcrossrefs} explicitly, but you may want to redefine it to only iterate over specific glossaries. The unused entries are indexed using the \texttt{glsxtrunusedformat} format.

\glsxtrunusedformat{⟨location⟩}

This ignores its argument (the location) and just does \texttt{\unskip}.

5.10. First Use Flag

Each entry has an associated first use flag (a conditional or boolean variable), which determines whether or not the entry has been marked as “used”. Unsetting this flag means that the entry is marked as used. Resetting the flag means that the entry is marked as unused.

The \texttt{\gls}-like commands (which are the principle method of referencing an entry) all mark the entry as used after the link text is displayed but before the post-link hook is used.
5. Referencing (Using) Entries

The purpose of this is to allow for additional information that needs to be shown when a term first appears in a document. For example, an abbreviation may need to have its full form shown on the instance. However, there are some cases where that additional information may need to be shown again or where the literal first instance of the term may need to be in its terse form. For example, if the term is used in the front matter.

If any \gls-like commands (which are robust) are used in section headings or captions, they can end up in the table of contents or corresponding “list of...” (such as the list of figures). This can cause the first use flag to be unset too soon. For these situations, use the commands described in §5.3 instead.

The base glossaries package provides commands to explicitly unset or reset the first use flag either locally (confined to the current scope) or globally. These commands are: \glsunset (global unset), \glslocalunset (local unset), \glsreset (global reset) and \glslocalreset (local reset).

The glossaries-extra package adds hooks to the above commands. These do nothing by default, but are modified by \glsenableentrycount and \glsenableentryunitcount to perform the count increment or reset (see §6.1).

\glsxtrpostunset\{⟨entry-label⟩\}

This hook is added to \glsunset.

\glsxtrpostlocalunset\{⟨entry-label⟩\}

This hook is added to \glslocalunset.

\glsxtrpostreset\{⟨entry-label⟩\}

This hook is added to \glsreset.

\glsxtrpostlocalreset\{⟨entry-label⟩\}

This hook is added to \glslocalreset.

The base package also provides commands to unset or reset all entries or all entries within particular glossaries: \glsunsetall and \glsresetall. For example, if you don’t want the first use in the front matter, you can unset all entries at the start of the front matter and reset them at the start of the main matter.
5. Referencing (Using) Entries

With glossaries-extra you can unset a specific subset of entries.

\glslocalunseteach\{⟨entry-labels⟩\}

Locally unsets each entry in the given comma-separated list of entry labels.

\glslocalreseteach\{⟨entry-labels⟩\}

Locally resets each entry in the given comma-separated list of entry labels.

You can test if an entry has been marked as used with \ifglsused (but take care if you are using bib2gls or the undefaction=warn option, see below). This command allows the entry display style to vary the link text according to whether or not the entry has been marked as used. However, it can’t be used within the post-link hook as by that time, the first use flag will have already been unset.

See the glossaries user manual for further details of the above commands.

For example, in the following document the “html” entry is first used in the abstract, which shows both the long and the short form, but it would be helpful for the full form to be reshow in the main section about web pages. This is achieved by resetting the first use flag.

\newabbreviation\{html\}\{HTML\}\{hypertext markup language\}
\begin\{document\}
  \begin\{abstract\}
  This abstract mentions \gls\{html\}.
  \end\{abstract\}
  Some casual reference to \gls\{html\}.
\section\{Web Pages\}
  \glsreset\{html\}This section is all about \gls\{html\}.
\end\{document\}
5. Referencing (Using) Entries

Example 113: Resetting the first use flag (\glsreset)

Abstract

This abstract mentions hypertext markup language (HTML).

Some casual reference to HTML.

1 Web Pages

This section is all about hypertext markup language (HTML).

In the above example, an alternative is to use \glsxtrfull where you particularly want the full form, but some abbreviation styles have a different expansion with the inline \glsxtrfull form compared with the first use of \gls.

The glossaries-extra package provides the options \texttt{preunset} and \texttt{prereset}, which can be used to unset or reset the first use flag before the link text. This means that in the above example, the line:

\begin{quote}
\glsreset{html}This section is all about \gls{html}.
\end{quote}

can be replaced with:

\begin{quote}
This section is all about \gls[prereset]{html}.
\end{quote}

As mentioned above, the first use flag is unset before the post-link hook, so \texttt{if\glsused} isn’t helpful in the post-link hook. Instead, you can use:

\begin{quote}
\glsxtrifwasfirstuse{⟨true⟩}{⟨false⟩}
\end{quote}

This command is initialised by the \texttt{gls}-like commands according to the value of the first use flag before the link text. It’s also initialised by the \texttt{glstext}-like commands: not according to the value of the first use flag but according to whether or not the \texttt{glstext}-like command emulates first use.

For example, \texttt{gls} will define \texttt{\glsxtrifwasfirstuse} to do its first argument if the first use flag indicates the entry hasn’t yet been used, otherwise it will define \texttt{\glsxtrifwasfirstuse} to do its second argument. Whereas \texttt{glsfirst} will always define \texttt{\glsxtrifwasfirstuse} to do its first argument (unless used with \texttt{preunset}) and \texttt{glstext} will always define \texttt{\glsxtrifwasfirstuse} to do its second argument (unless used with \texttt{prereset}), regardless of the state of the first use flag.
The `preunset` and `prereset` options will additionally redefine `\glsxtrifwasfirstuse` to match the option. See §5.5.4 for further details about the post-link hook.

If you want to check if the calling command was both the first use and it was a `\gls`-like command, you can use: `\glsxtrifwasglslikeandfirstuse`.

The unset function performed by the `\gls`-like commands before the post-link hook uses the global `\glsunset` by default. If you want `\glslocalunset` instead, you can use the `local` option (provided by the base glossaries package) or `postunset=local`. To prevent the first use flag from being unset after the link text, use `postunset=none`.

```latex
\newabbreviation{html}{HTML}{hypertext markup language}
\begin{document}
\% local scope
\gls[local]{html}. Used? \ifglsused{html}{Yes}{No}.
\% end scope
Used? \ifglsused{html}{Yes}{No}.
\gls[postunset=none]{html}. Used? \ifglsused{html}{Yes}{No}.
\end{document}
```

Example 114: Local unset

hypertext markup language (HTML). Used? Yes.
Used? No.

If you are using the `undefaction=warn` option (which is automatically implemented by the `record` option), the first use flag is undefined and so is neither true nor false, in which case `\ifglsused` will trigger an error or warning and do neither. In this situation, you may need to use the following command instead.

```latex
\GlsXtrIfUnusedOrUndefined{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}
```

This does ⟨true⟩ if the entry hasn’t been defined or hasn’t been marked as used, otherwise does ⟨true⟩. Note that this command will generate an error or warning (according to `undefaction`) if the entry hasn’t been defined, but will still do ⟨true⟩. This is more useful than `\ifglsused` with `bib2gls` where the entries are never defined on the first \LaTeX run.
5.10.1. Buffering Unsets

Sometimes commands like \gls are used in a context where changing a boolean variable can cause things to go wrong. The outer, middle and inner formatting (see §5.5) can be used to change the font for the link text, but it may be that the \gls-like command occurs within a block of text that needs to be encapsulated by such a command.

One example of this is using \gls in one of the commands provided with the soul package. For example:

\ul{Some text about \gls{html}.}

This causes the confusing error:

Glossary entry `{html}' has not been defined.

The simplest workaround is to put \gls{html} inside the argument of \mbox. For example:

\ul{Some text about \mbox{\gls{html}}.}

This can work provided it’s not the first use of this entry. If it is, then unsetting the first use flag causes a problem and results in the error:

! Package soul Error: Reconstruction failed.

The glossaries-extra package provides a way of temporarily switching off \glsunset so that it just makes a note of the entry’s label but doesn’t actually perform the change.

\GlsXtrStartUnsetBuffering

This starts the buffering. The unstarrred version doesn’t check for duplicates, so the internal list may end up with multiple occurrences of the same label. The starred version only adds a label to the internal list if it’s not already in it. If you are using entry counting (see §6.1) the unstarrred version is preferable to ensure the entry count is correct.

Buffering only applies to the global \glsunset and does not affect the local \glslocalunset.
The buffer can be locally cleared with:\GlsXtrClearUnsetBuffer
This doesn’t stop buffering. It will simply discard the labels that have been buffered so far.
In order to restore the normal behaviour of \glsunset, the buffering must be stopped or discarded.
\GlsXtrStopUnsetBuffering \textit{modifier: *}
This stops the buffering, restores \glsunset, and unset all the buffered labels. The starred form uses \glslocalunset to unset the buffered labels.
Before you stop the unset buffering, you can iterate over the current buffer.
\GlsXtrForUnsetBufferedList{\langle handler-cs\rangle}
This iterates over the currently buffered list of entry labels and performs \langle handler-cs\rangle\{(entry-label)\} for each label, where \langle cs\rangle is a control sequence that takes a single argument. This is best used with the starred version of \GlsXtrStartUnsetBuffering* to avoid duplicates.
\GlsXtrDiscardUnsetBuffering
This discards the buffer and restores \glsunset to its normal behaviour.
It’s possible to locally unset entries before use (analogous to \texttt{preunset=local}) if the entry has already been encountered in the buffer. This will still be problematic for situations where changing a conditional causes a problem, but may be useful in some situations. This feature is enabled with:\GlsXtrUnsetBufferEnableRepeatLocal
This may be placed before or after \GlsXtrStartUnsetBuffering but the locally collected list of unused entries will be cleared at the start of each instance of \GlsXtrStartUnsetBuffering. It will also be cleared by \GlsXtrClearUnsetBuffer. All entries that have been marked as unused can be reset with:\GlsXtrResetLocalBuffer
This will perform a local reset on all the entries in the “not used” list and do \GlsXtrClearUnsetBuffer.

This feature can be switched off with:

\GlsXtrUnsetBufferDisableRepeatLocal

It’s disabled by default.

The way this feature works is as follows (while buffering is active):

1. Each time an entry is referenced with a \gls-like command, the \glsinitreunsets hook checks if the current entry (identified by \glslabel) has been added to the buffer. (Bear in mind that the label is added to the buffer after the link text when \glsunset is used.)
   a) If it has been added to the buffer, then this is an indication that the entry has already been used within the buffer zone (that is, an attempt has been made to globally unset the first use flag). A local unset is then performed, which is essentially equivalent to using the \texttt{preunset=local} option, so the reference will behave like subsequent use.
   b) If it hasn’t been added to the buffer, then this is an indication that the entry hasn’t already been used within the buffer zone, but it may or may not have been used before the buffering started. If the first use flag indicates that the entry hasn’t been used, the entry’s label will be added to the “not used” list. The reference will behave like first use, but the unset won’t be performed afterwards (because buffering is in progress).

2. The entries that are in the “not used” list can be locally reset and both the buffer and the “not used” list can be cleared with \GlsXtrClearUnsetBuffer.

Note that this approach can’t be used for situations where the change in conditional causes a problem, but it can be used in situations where the content of an environment or command is repeatedly processed, which upsets the first use flag.

For example, consider the following beamer document:

\documentclass{beamer}
\usepackage{glossaries-extra}
\newabbreviation{svm}{SVM}{support vector machine}
\newabbreviation{html}{HTML}{hypertext markup language}
\begin{document}
\begin{frame}
\frametitle{Frame 1}
\begin{itemize}
\item \gls{html} and \gls{html}
\end{itemize}
\end{frame}
\end{document}
The first page isn’t a problem as the frame doesn’t have overlays. The first reference of the “html” entry shows the full form and the next shows just the short form. The second page (which is the first of the overlays of the second frame) correctly shows the full form of the “svm” entry for the first reference and the short form for the second reference, but on the third page (the second of the overlays) now has all instances of “svm” showing as subsequent use (just the short form).

I could put \glslocalresetall at the start of the frame, but that would reset the “html” entry as well. Another workaround is to locally reset the first “svm” entry with \preresetlocal, but that defeats the point of the first use flag, which is intended to keep track of whether or not you have used an entry so that you don’t have to.

The frame can be placed inside a buffering zone:

\GlsXtrStartUnsetBuffering
\begin{frame}
\frametitle{Frame 2}
\begin{itemize}
\item<+-> \gls{svm} and \gls{svm}
\item<+-> \gls{svm} and \gls{html}
\end{itemize}
\end{frame}
\GlsXtrStopUnsetBuffering

This ensures that the first use flag isn’t reset until after the frame, but it means that all references to the “svm” entry on both the second and third page show the full form.

The “repeat local” function can be used so that repeated references for the same entry can be locally unset before use. This can be enabled with \GlsXtrUnsetBufferEnableRepeatLocal which fixes the second page, but not the third page, which shows all references to “svm” as the short form. What’s needed is to locally reset and entries that are in the frame but haven’t yet been used (“svm”, in this case) at the start of the frame with \GlsXtrResetLocalBuffer:
5. Referencing (Using) Entries

\begin{frame}
\frametitle{Frame 2}
\begin{itemize}
\item<+-> \gls{svm} and \gls{svm}
\item<+-> \gls{svm} and \gls{html}
\end{itemize}
\end{frame}

Note that on the first overlay, the buffer and “not used” list are both empty. On the second overlay, the buffer contains the “svm” and “html” labels and the “not used” list just contains the “svm” label. The reset performed by \GlsXtrResetLocalBuffer will reset “svm” and then clear both the buffer and the “not used” list. This means that the first “svm” reference is once again considered first use and it will once again be added to the “not used” list (so that it would be reset again if there was a third overlay).

Example 115: Abbreviations with beamer (unset buffering)

This is quite cumbersome, but these commands could potentially be added to hooks at the start and end of problematic environments (but the buffering needs to be started and ended outside of the repeated content).

The following example uses \mbox to protect \gls within the buffer zone:

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{soul}
\usepackage{glossaries-extra}
\newabbreviation{html}{HTML}{hypertext markup language}
5. Referencing (Using) Entries

This produces:

Example 116: Buffering first use unsets with \mbox

Some text about hypertext markup language (HTML). Next use: hypertext markup language
Next use: HTML.

Note that the use of \mbox prevents line-breaking and the second instance of \gls{html} is treated as first use.

Note that since the change in the first use flag now doesn’t occur until \GlsXtrStopUnsetBuffering, multiple references of the same term within the buffering zone will always be treated as first use (if the term wasn’t used before the buffering started).

Other alternatives include using \protect and inner formatting (see §5.5.3 for limitations) or middle formatting (see §5.5.2) with \GlsXtrExpandedFmt (which can’t be used with fragile link text). Both approaches are demonstrated in the following example:
5. Referencing (Using) Entries

\renewcommand{\glsxtrdefaultentrytextfmt}{\ul{#1}}\
\ul{Some text about \protect\gls{html}.}
Next use: \protect\gls{html}}
\}

Next use: \gls{html}.

Second approach (middle formatting with expanded link text):
\glsresetall
{% scope
\renewcommand{\glsxtrabbreviationfont}{\xpul{#1}}%
\renewcommand{\glsxtrregularfont}{\xpul{#1}}%
\ul{Some text about \protect\gls{html}.}
Next use: \gls{html}.}
\}

Next use: \gls{html}.
\end{document}

This produces:

Example 117: Alternatives to buffering

First approach (inner formatting): Some text about hypertext markup
language (HTML). Next use: HTML
Next use: HTML.
Second approach (middle formatting with expanded link text): Some text
about hypertext markup language (HTML). Next use: HTML.
Next use: HTML.

The change in the first use flag isn’t the only content within the \gls-like commands that
can cause a problem. The whatsit caused by indexing can also be problematic. Buffering
can also be used to help with that situation. Indexing can be switched off at the start of the
buffering and \GlsXtrForUnsetBufferedList can be used to perform the indexing outside
of the problematic content. Note that this can cause a problem if the location changes (for
example, if a page break occurs within the buffering zone).

Buffering can also be used to simply gather the labels that have been referenced with a
\gls-like command in order to, for example, display a mini-glossary at the end of the block.
See for example, Gallery: Mini-Glossary.\footnote{dickimaw-books.com/gallery/index.php?label=minigloss}

\footnote{dickimaw-books.com/gallery/index.php?label=minigloss}
5.11. Accessing Fields

See §3.5 for setting fields after an entry has been defined, §5.13 for fields that contain comma-separated lists or whose values may be contained within comma-separated lists, §5.9.2 for cross-referencing fields (see, seealso and alias), and §5.15 for testing field values. See also the base glossaries package’s commands, such as \glsentryname and \glsletentryfield.

\glsxtrusefield{⟨entry-label⟩}{⟨field-label⟩}

This expands to the value of the field (identified by its internal label ⟨field-label⟩) for the entry identified by ⟨entry-label⟩. Expands to \relax if the field or entry are undefined.

\Glsxtrusefield{⟨entry-label⟩}{⟨field-label⟩}

This is like \glsxtrusefield but converts the first character to uppercase using \makefirstuc (provided by mfirstuc) which is robust. If hyperref is loaded, \Glsxtrusefield{⟨entry-label⟩} will use the expandable:

\MFUsentencecase{\glsxtrusefield{⟨entry-label⟩}}

in a PDF bookmark.

\GLSxtrusefield{⟨entry-label⟩}{⟨field-label⟩}

This is like \glsxtrusefield but converts the field value to uppercase. See §5.2.3.

\glsxtrfieldtitlecase{⟨entry-label⟩}{⟨field-label⟩}

This is like \glsxtrusefield but converts the field value to title case. This internally uses:

\glsxtrfieldtitlecases{⟨content⟩}

This converts ⟨content⟩ to title case (expanding the first token once). If \glscapitalisewords has been defined, that will be used, otherwise \capitalisewords will be used.

\glsxtrentryparentname{⟨entry-name⟩}
5. Referencing (Using) Entries

Expands to the entry’s parent name if defined. Expands to nothing if the entry doesn’t have a parent or if the entry isn’t defined. If you simply require the parent label then use \glsentryparent or, to first test if the entry has a parent, either use \ifglshasparent or use \glsxtrifhasfield with the field label set to parent.

\glsxtrhiername{(entry-label)}

Displays the hierarchical name for the given entry. The cross-reference format \glsseeitemformat may be redefined to use this command to show the hierarchy, if required.

This command has a recursive definition. If the entry given by \langle entry-label \rangle has a parent, then this command will do \glsxtrhiername{(parent-label)} for the entry’s parent and will then do the separator \glsxtrhiernamesep.

Then, regardless of whether or not the entry has a parent, it will do \glsfmttext{(entry-label)}, if the entry is an abbreviation (see §1.2.4), or \glsfmtname{(entry-label)} otherwise.

If hyperref is loaded, \glsxtrhiername will behave as \glsentryname in a PDF bookmark.

\glsxtrhiernamesep

Separator symbol (⊿) used between each name in commands like \glsxtrhiername.

\Glsxtrhiername{(entry-label)}

As \glsxtrhiername but the first name in the list has its first character converted to uppercase using \Glsfmttext or \Glsfmtname (sentence case). If hyperref is loaded, \Glsxtrhiername will expand to:

\MFUsentencecase{\glsentryname{(entry-label)}}

in a PDF bookmark. The \makefirstuc mapping from \glsxtrhiername to \Glsxtrhiername is set with \glsmfuaddmap, if supported.

\GlsXtrhiername{(entry-label)}

As \glsxtrhiernname but each name in the list has its first character converted to uppercase
5. Referencing (Using) Entries

using \Glsfmttext or \Glsfmtname.

\GLSEXTRhiername{⟨entry-label⟩}

As \glsxtrhiername but the first name in the list is converted to uppercase using \GLSfmttext or \GLSfmtname.

\GLSXTRhiername{⟨entry-label⟩}

As \glsxtrhiername but each name in the list is converted to uppercase using \GLSfmttext or \GLSfmtname (all caps).

5.12. Encapsulation (Formatting) Based on Field Values

These commands assume that a given entry has a special purpose field that’s used to store information on how to format text.

5.12.1. Foreign Language Field

\GlsXtrForeignTextField

This command should expand to the internal field label used to store a language tag (such as en-GB or de-CH-1996). The default value is userii (which corresponds to the user2 key).

\GlsXtrForeignText{⟨entry-label⟩}{⟨text⟩}

If the entry given by ⟨entry-label⟩ has the field identified by \GlsXtrForeignTextField set, then this command will encapsulate ⟨text⟩ according to the language tag stored in that field.

This uses tracklang’s interface to determine the language label that corresponds to the language tag. If the language label can be determined, the ⟨text⟩ will be encapsulated with \foreignlanguage otherwise just ⟨text⟩ is done.

If \foreignlanguage isn’t defined (that is, there’s no language support for the document), this command simply does ⟨text⟩. If an old version of tracklang is used, this command issues a warning and just does ⟨text⟩.

If tracklang can’t determine the corresponding language label to use with \foreign-
5. Referencing (Using) Entries

language, then a warning is issued with:

\GlsXtrUnknownDialectWarning{\langle \text{locale} \rangle}{\langle \text{root language} \rangle}

where \langle \text{locale} \rangle is the language tag supplied in the given field value and \langle \text{root language} \rangle is the root language that \text{tracklang} has inferred from the tag.

\GlsXtrForeignText requires \text{tracklang} v1.3.6+.

For example:

\usepackage[main=british,brazilian,ngerman]{babel}
\usepackage{glossaries-extra}
\setabbreviationstyle{long-short-user}
\newabbreviation{abnt}{ABNT}{Brazilian National Standards Organization}
\newabbreviation{din}{DIN}{German Institute for Standardization}
\newabbreviation{tug}{TUG}{\TeX\ User Group}
\renewcommand*{\glsxtruserparen}[2]{%
  \glsxtrfullsep[#2]\
  \glsxtrparen
  \ifgls\hasfield{\glsxtruserfield}{#2}{%
    , \textbf{\GlsXtrForeignText[#2]{\glscurrentfieldvalue}}%
  %
  }%
}
\begin{document}
\gls{abnt}, \gls{din}, \gls{tug}.
\printunsrtglossaries

297
5. Referencing (Using) Entries

This produces:

Example 118: Foreign language field encapsulation

Brazilian National Standards Organization (ABNT, Associação Brasileira de Normas Técnicas), German Institute for Standardization (DIN, Deutsches Institut für Normung e.V.), T\TeX User Group (TUG).

Glossary

\textbf{ABNT}  Brazilian National Standards Organization

\textbf{DIN}  German Institute for Standardization

\textbf{TUG}  T\TeX User Group

5.12.2. Associated Entry Format

An entry may have a particular formatting style associated with it (rather than a more general category-wide format). This needs to be provided by a text-block command that takes a single argument. The name (without the leading backslash) should be stored in the field identified by:

\begin{verbatim}
\GlsXtrFmtField initial: useri
\end{verbatim}

This command should expand to the internal field label used to store the formatting command’s control sequence name. The default value is \texttt{useri} (which corresponds to the \texttt{user1} key).

\begin{verbatim}
\glsxtrfmt\{\langle\text{options}\rangle\}\{\langle\text{entry-label}\rangle\}\{\langle\text{text}\rangle\}
\end{verbatim}

This command behaves like:

\begin{verbatim}
\glslnk\{\langle\text{options}\rangle\}\{\langle\text{entry-label}\rangle\}\{\langle\text{fmt-link-text}\rangle\}
\end{verbatim}
5. Referencing (Using) Entries

where the link text \langle fmt-link-text \rangle is formatted according to:

\texttt{\glsxtrfmtdisplay{(csname)}{(text)}{(insert)}}

The default definition simply does \langle csname \rangle \langle text \rangle \langle insert \rangle where the control sequence name \langle csname \rangle is obtained from the field given by \texttt{\GlsXtrFmtField}. If the field hasn’t been set, \texttt{@firstofone} is used (which simply does its argument). The unstarrred version assumes an empty \langle insert \rangle. The default \texttt{\glslink} options are given by \texttt{\GlsXtrFmtDefaultOptions}.

The post-link hook is suppressed with \texttt{\glsxtrfmt}.

If you don’t want the complexity of \texttt{\glslink}, a partially expandable command is provided that may be used in section headings:

\texttt{\glsxtrentryfmt{(entry-label)}{(text)}}

If hyperref has been loaded, this will expand to:

\texttt{\glsxtrpdfentryfmt{(entry-label)}{(text)}}

within the PDF bookmarks, which just does \langle text \rangle. Otherwise \texttt{\glsxtrentryfmt} will format \langle text \rangle according to the control sequence name identified in the field given by \texttt{\GlsXtrFmtField} (or \texttt{@firstofone}, if not set). For example:

\texttt{\documentclass{article}
  \usepackage[T1]{fontenc}
  \usepackage{amsmath}
  \usepackage[colorlinks]{hyperref}
  \usepackage[postdot,style=index]{glossaries-extra}
  \makeglossaries
  \newcommand*{\mtx}{\boldsymbol{#1}}
  \newcommand*{\mtxinv}{\mtx{#1}\sp{-1}}

  \newglossaryentry{matrix}{%
    name={matrix},
    symbol={\ensuremath{\mtx{M}}},
    plural={matrices},
    user1={\mtx},% corresponds to \mtx
  }
5. Referencing (Using) Entries

\newglossaryentry{identitymatrix}{%  
  name={identity matrix},  
  symbol={\ensuremath{\mtx{I}}},  
  plural={identity matrices},  
  description={a diagonal matrix with all diagonal elements equal to 1 and all other elements equal to 0}}

\newglossaryentry{matrixinv}{%  
  name={matrix inverse},  
  symbol={\ensuremath{\mtxinv{M}}},  
  user1={\_mtxinv},% corresponds to \mtxinv  
  description={a square \gls{matrix} such that \$\mtx{M}\mtxinv{M}={\glssymbol{identitymatrix}}\$}}

\begin{document}
A \gls{matrix} is denoted \glssymbol{matrix}. The inverse is denoted \glssymbol{matrixinv}.\[
\glsxtrfmt{matrix}{A} \glsxtrfmt{matrixinv}{A} = \glssymbol{identitymatrix}
\]
Compare $\glsxtrfmt{matrix}{A}[\_0]$ with $\glsxtrfmt*{matrix}{A}[\_0]$. \printglossaries
\end{document}

This produces:
Example 119: Storing a formatting command in a field

A matrix is denoted \( M \). The inverse is denoted \( M^{-1} \).

\[ AA^{-1} = I \]

Compare \( A_0 \) with \( A_0 \).

**Glossary**

- **identity matrix** \( (I) \) a diagonal matrix with all diagonal elements equal to 1 and all other elements equal to 0. 1
- **matrix** \( (M) \) rectangular array of values. 1
- **matrix inverse** \( (M^{-1}) \) a square matrix such that \( MM^{-1} = I \). 1

Note the difference between using \glsxtrfmt* vs \glsxtrfmt. There are also sentence case versions of the above commands:

\begin{verbatim}
\glsxtrfmt{⟨options⟩}{⟨entry-label⟩}{⟨text⟩}
\end{verbatim}

This is simply a shortcut for:

\begin{verbatim}
\glsxtrfmt{⟨options⟩}{⟨entry-label⟩}{\glssentencecase{⟨text⟩}}
\end{verbatim}

Similarly for the starred form:

\begin{verbatim}
\glsxtrfmt*{⟨options⟩}{⟨entry-label⟩}{⟨text⟩}{⟨insert⟩}
\end{verbatim}

which is a shortcut for:

\begin{verbatim}
\glsxtrfmt*{⟨options⟩}{⟨entry-label⟩}{\glssentencecase{⟨text⟩}}{⟨insert⟩}
\end{verbatim}

\begin{verbatim}
\Glsxtrentryfmt{⟨entry-label⟩}{⟨text⟩}
\end{verbatim}

This is a shortcut for
5. Referencing (Using) Entries

\begin{Verbatim}
\glsxtrentryfmt{⟨entry-label⟩}\{\glssentencecase{⟨text⟩}\}
\end{Verbatim}

but uses:

\begin{Verbatim}
\Glsxtrpdfentryfmt{⟨entry-label⟩}\{⟨text⟩\}
\end{Verbatim}

for the PDF bookmarks. This uses \texttt{\MFUsentencecase} to perform the case-change, which is expandable.

If you are writing \texttt{\glsxtrfmt} or \texttt{\glsxtrentryfmt} explicitly in the document text, you can, of course, enter the appropriate case in \texttt{⟨text⟩} directly. The purpose of providing the sentence case commands is to enable a mapping to be setup with \texttt{\MFUaddmap} in the event that \texttt{\glsxtrfmt} or \texttt{\glsxtrentryfmt} occur at the start of content, such as another entry’s description, that will have sentence case automatically applied. This will require \texttt{mfirstuc v2.08+} to support the mapping. See the \texttt{mfirstuc} manual for further details.

5.13. Comma-Separated Lists

These commands are for field values that are comma-separated lists (for example, the field has been constructed with \texttt{\glsxtrapptocsvfield}) or for testing if field values are contained within comma-separated lists.

If you are using \texttt{bib2gls}, you can sort field values that contain a comma-separated list of labels (such as the \texttt{see} or \texttt{seealso} field) with the \texttt{sort-label-list} option (provided \texttt{bib2gls} can access those fields). See the \texttt{bib2gls} manual for further details.

\begin{Verbatim}
\glsseelist{⟨csv-list⟩}
\end{Verbatim}

This is provided by the base glossaries package to format the entry labels in \texttt{see} cross-reference list. (It’s used internally by \texttt{\glsseeformat}, which adds the \texttt{see} prefix.) It may also be used for any comma-separated list of entry labels. Note that the argument isn’t expanded. If expansion is required, use:

\begin{Verbatim}
\glsxtrseelist{⟨csv-list⟩}
\end{Verbatim}

This fully expands its argument and passes the result to \texttt{\glsseelist}. With just the base glossaries package, each item is encapsulated with \texttt{\glsseeitem}. The glossaries-extra package redefines \texttt{\glsseelist} to make it more flexible and provides additional commands to
5. Referencing (Using) Entries

further customize the formatting.

\glsxtrtaggedlist{(singular tag)}{(plural tag)}{(label prefix)}{(csv-list)}

This is a similar command that has an initial tag inserted before the start of the list. If the list only contains one element, the (singular tag) is used. If the list contains more than one element, the (plural tag) is used. The separator between the tag and the list is given by:

\glsxtrtaggedlistsep

initial: \space

The separators between the elements of the list and the formatting of each list element is as for \glsseelist (see below). If the list is empty, nothing is displayed. The (label prefix) is inserted before the current item in the list to form the entry label.

Spaces in (csv-list) are significant. Avoid unwanted leading or trailing spaces and empty labels.

\glsseeitemformat{(entry-label)}

The base glossaries package just uses \glsentryname or \glsentrytext in this command. The glossaries-extra package redefines this so that it does:

\ifglshasshort{(entry-label)}{\glsfmttext{(entry-label)}}{\glsfmtname{(entry-label)}}

Note that the use of \glsfmttext rather than \glsentrytext allows the abbreviation style to be used.

With glossaries-extra, the first item in \glsseelist will be encapsulated with:

\glsseefirstitem{(entry-label)}

The default definition is simply \glsseeitem{(entry-label)} but can be redefined, for example to convert the first character to uppercase if sentence case is required.
If the label corresponds to a multi-entry, \texttt{\glsseefirstitem} will be used instead (see §7.12). Similarly, \texttt{\glsseeitem} will be used instead of \texttt{\glsseeitem} for a multi-entry label.

\begin{itemize}
\item \texttt{\glsseesep}
\end{itemize}

This is used between each entry in the list, except between the final pair.

\begin{itemize}
\item \texttt{\glsseelastsep}
\end{itemize}

This is used between the penultimate and final item in the list. The default definition is:

\begin{itemize}
\item \texttt{\space \andname \space}
\end{itemize}

(\texttt{\andname} is provided by glossaries, if not already defined, and simply expands to \texttt{\&} but it may be defined to expand to something else by another package before glossaries is loaded.)

With glossaries-extra, if there are at least three elements in the list, the separator between the final two elements will be given by:

\begin{itemize}
\item \texttt{\glsseelastoxfordsep}
\end{itemize}

This just defaults to \texttt{\glsseelastsep} but may be redefined to include a comma, if preferred.

\begin{itemize}
\item \texttt{\glsxtrforcsvfield\{\langle entry-label\rangle\}\{\langle field-label\rangle\}\{\langle handler cs\rangle\}} \quad \text{modifier: } ^*\quad \text{(modifier: } ^*\text{)}
\end{itemize}

This iterates over the comma-separated list stored in the given field (identified by its internal label) for the entry identified by \texttt{\langle entry-label\rangle} and performs \texttt{\langle handler cs\rangle\{\langle element\rangle\}} for each element of the list. This command uses \texttt{\glsxtrifhasfield} so the complete list can be obtained with \texttt{\glscurrentfieldvalue}. Does nothing if the field hasn’t been set or the entry hasn’t been defined. The unstarred version adds implicit grouping. The starred version doesn’t.

It’s possible to prematurely break the loop at the end of the current iteration with:

\begin{itemize}
\item \texttt{\glsxxtrendfor}
\end{itemize}
5. Referencing (Using) Entries

If nested within another command that also uses `@for`, use the unstarred version to localise the break. This command is simply set to `@endfortrue`, which is provided by the xfor package.

\glsxtrfieldformatcsvlist\{⟨entry-label⟩\}\{⟨field-label⟩\}

This formats the comma-separated list stored in the given field (identified by its internal label) for the entry identified by `⟨entry-label⟩` using datatool-base’s `DTLformatlist`. This command uses `\glsxtrifhasfield` so the complete list can be obtained with `\glscurrentfieldvalue`. This adds implicit grouping. There is no starred version.

The following demonstrates the difference between `\glsseelist` (which specifically requires a list of labels) and `\glsxtrfieldformatcsvlist` (which formats an arbitrary list):

\usepackage[colorlinks]{hyperref}
\usepackage[autoseeindex=false]{glossaries-extra}
\newglossaryentry{example}{name={example},description={},see={another1,another2}}
\newglossaryentry{another1}{name={another one},description={}}
\newglossaryentry{another2}{name={another two},description={}}
\begin{document}
\glsxtrapptocsvfield{example}{animals}{duck}
\glsxtrapptocsvfield{example}{animals}{albatross}
\glsxtrapptocsvfield{example}{animals}{arara}
Animal list: \glsxtrfieldformatcsvlist{example}{animals}

See list: \glsxtrifhasfield{see}{example}
\{\glsseelist{\glscurrentfieldvalue}\{not set\}.
\printunsrtglossaries
\end{document}

There’s no indexing in this document so I’ve used `autoseeindex=false` to avoid an error. This means there’s no cross-reference list in the glossary but, as demonstrated, the “see” list can be reproduced in the document. The result is:
Example 120: Formatting lists contained in field values

Animal list: duck, albatross & arara.
See list: another one & another two.

Glossary

example
another one
another two

This first constructs a comma-separated list in a custom internal field with the label animals. There’s no associated key that can be used in \newglossaryentry. In this case, the field could simply be set in one command. For example:

\glsxtrdeffield{example}{animals}{duck,albatross,arara}

The main reason for providing \glsxtrapptocsvfield is for the benefit of bib2gls, as it sometimes has to construct a field value list while it’s writing the glstex file, but there may be other uses in complex documents that construct field values through some custom function.

\GlsXtrIfValueInFieldCsvList{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}

This does ⟨true⟩ if the comma-separated list stored in the given field (identified by its internal label) contains the given ⟨value⟩ (using DTLifinlist provided by datatool-base) or ⟨false⟩ if the value isn’t in the list or if the field hasn’t been set or the entry hasn’t been defined. The unstarred version adds implicit grouping. The starred version doesn’t.

This command internally uses \glsxtrifhasfield, so take care if it’s nested. Within ⟨false⟩, you can test if \glscurrentfieldvalue is empty or undefined. If it’s defined but not empty, then the field has been set but doesn’t contain ⟨value⟩.

\GlsXtrIfFieldValueInCsvList{⟨entry-label⟩}{⟨field-label⟩}{⟨csv-list⟩}{⟨true⟩}{⟨false⟩}

This command is essentially the other way around to the above. In this case, the comma-
Referencing (Using) Entries

A separated list is provided in the argument ⟨csv-list⟩ and the search value is the field’s value. This does ⟨true⟩ if the value is found in ⟨csv-list⟩ or ⟨false⟩ if the value isn’t in ⟨csv-list⟩ or the field isn’t set or the entry hasn’t been defined. The unstarred version adds implicit grouping. The starred version doesn’t. Again, this command internally uses \glsxtrifhasfield, so you can test \glscurrentfieldvalue in ⟨false⟩ to determine whether or not the field has been set.

\xGlsXtrIfValueInFieldCsvList{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}

As \GlsXtrIfValueInFieldCsvList but fully expands ⟨value⟩ first.

5.14. List Fields

Comma-separated list fields are covered in §5.13. The commands in this section are for fields that store etoolbox internal lists. Elements can be appended to these fields using commands \glsxtrfieldlistadd, described in §3.5. The commands listed below provide an easy interface to iterate over the field values. See the etoolbox documentation for further details about internal lists.

\glsxtrfieldformatlist{⟨entry-label⟩}{⟨field-label⟩}

Formats the list using the same separators as used by datatool’s \DTLformatlist. This internally uses etoolbox’s \forlistcsloop with the same handler macro as used with \DTLformatlist.

\glsxtrfielddolistloop{⟨entry-label⟩}{⟨field⟩}

This uses etoolbox’s \dolistcsloop, which uses the command \do as the handler.

\glsxtrfieldforlistloop{⟨entry-label⟩}{⟨field⟩}{⟨handler-cs⟩}

This uses etoolbox’s \forlistcsloop, which uses the ⟨handler-cs⟩ as the handler.

\glsxtrfieldifinlist{⟨entry-label⟩}{⟨field⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}
This uses etoolbox's \ifinlistcs to test if ⟨item⟩ is in the list.

\glsxtrfieldxifinlist{⟨entry-label⟩}{⟨field⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}

This uses etoolbox's \xifinlistcs to test if ⟨item⟩ is in the list.

5.15. Field Conditionals

\GlsXtrIfFieldUndef{⟨field-label⟩}{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}

Tests if the given field (identified by its internal label) is undefined for the entry given by ⟨entry-label⟩. Does ⟨true⟩ if the entry doesn't exist or if entry exists but the field hasn't been set. Does ⟨false⟩ if the field has been set, even if it has been set to empty. Unlike \glsxtrifhasfield there is no grouping or starred version and no assignment of \glscurrentfieldvalue. This is simply a shortcut that internally uses etoolbox's \ifcsundef. The base glossaries package provides a similar command \ifglsfieldvoid, which uses etoolbox's \ifcsvoid instead.

\glsxtrifhasfield{⟨field-label⟩}{⟨entry-label⟩}{⟨true⟩}{⟨false⟩} modifier: *

This tests if the entry given by ⟨entry-label⟩ has the field identified by its internal label ⟨field-label⟩ set. This is like \ifglshasfield but doesn't produce a warning if the entry or field doesn't exist.

This command first assigns \glscurrentfieldvalue to the field value. If this is defined and not empty, ⟨true⟩ is done otherwise ⟨false⟩ is done. You can test \glscurrentfieldvalue within ⟨false⟩ to find out whether it's undefined or empty using etoolbox's commands, such as \ifundef or \ifdefempty.

The unstarred version adds implicit grouping to make nesting easier. The starred version doesn't (to make assignments easier).

If you are simply displaying the value of the field (for example, in the post-description hook) then use the unstarred version. If you are making an assignment based on the value of the field, then use the starred version.

\GlsXtrIfFieldCmpNum{⟨field-label⟩}{⟨entry-label⟩}{⟨op⟩}{⟨number⟩}{⟨true⟩}{⟨false⟩} modifier: *

308
5. Referencing (Using) Entries

This command should only be used with fields that contain integer values. It internally uses \texttt{\textbackslash glsXtrIfHasField} (the starred or unstarred version, to match the starred or unstarred version of \texttt{\textbackslash GlsXtrIfFieldEqStr}) and tests if \texttt{\textbackslash glsCurrentFieldValue} is equal to (\langle \textit{op} \rangle is \texttt{=}), less than (\langle \textit{op} \rangle is \texttt{<}) or greater than (\langle \textit{op} \rangle is \texttt{>}) the given number \langle \textit{number} \rangle. If the field is empty or undefined, \texttt{\textbackslash glsCurrentFieldValue} will be set to 0. Remember that the unstarred version adds implicit grouping.

\begin{verbatim}
\GlsXtrIfFieldEqNum{(field-label)}{(entry-label)}{(number)}{(true)}{(false)}
\end{verbatim}

This is a shortcut that uses \texttt{\textbackslash GlsXtrIfFieldCmpNum} with \langle \textit{op} \rangle set to \texttt{=}. The unstarred version adds implicit grouping.

\begin{verbatim}
\GlsXtrIfFieldNonZero{(field-label)}{(entry-label)}{(true)}{(false)}
\end{verbatim}

This is a shortcut that uses \texttt{\textbackslash GlsXtrIfFieldCmpNum} with \langle \textit{op} \rangle set to \texttt{=} and the final two arguments swapped. (So it’s true if the field value is not zero.) The unstarred version adds implicit grouping.

\begin{verbatim}
\GlsXtrIfFieldEqStr{(field-label)}{(entry-label)}{(value)}{(true)}{(false)}
\end{verbatim}

This internally uses \texttt{\textbackslash glsXtrIfHasField} (the starred or unstarred version, to match the starred or unstarred version of \texttt{\textbackslash GlsXtrIfFieldEqStr}) and tests if \texttt{\textbackslash glsCurrentFieldValue} is equal to \langle \textit{value} \rangle. Remember that the unstarred version adds implicit grouping.

\begin{verbatim}
\GlsXtrIfFieldEqXpStr{(field-label)}{(entry-label)}{(value)}{(true)}{(false)}
\end{verbatim}

This is like \texttt{\textbackslash GlsXtrIfFieldEqStr} but expands the string before the comparison. This also has an starred version that doesn’t add implicit grouping.

\begin{verbatim}
\GlsXtrIfXpFieldEqXpStr{(field-label)}{(entry-label)}{(value)}{(true)}{(false)}
\end{verbatim}

This is like \texttt{\textbackslash GlsXtrIfFieldEqStr} but expands both the field value and the string before the comparison. This also has an starred version that doesn’t add implicit grouping.
6. Counting References

There are three basic ways of counting entry references:

1. Counting the total number of times \texttt{glsunset} is used (\texttt{glsreset} resets the count unless \texttt{glsresetcurrcountfalse} and is best avoided). This is provided by the base glossaries package and is intended for documents where the term should be displayed differently if it’s only been used a certain number of times. The information has to be written to the aux file so that it’s available on the next \LaTeX{} run.

   This method is extended by glossaries-extra and is described in §6.1. This method relies on the document only using the \texttt{gls}-like commands and is inappropriate with \texttt{bib2gls}.

2. Counting the total number of records. This method is only available with \texttt{bib2gls} and is intended for documents where the term should be displayed differently if it’s only been recorded (indexed) a certain number of times. This is a more efficient method than entry counting. See §11.4 for further details.

3. Counting the number of times the \texttt{gls}-like or \texttt{glstext}-like commands are used. Unlike the other two methods, this just provides a running total rather than the total from the previous \LaTeX{} run. This method is intended to make it more convenient to work with hooks like \texttt{glslinkcheckfirsthyperhook}, \texttt{glslinkpostsetkeys} or \texttt{glslinkpresetkeys}. See §6.2 for further details.

6.1. Entry Counting (First Use Flag)

If you are using \texttt{bib2gls}, you need to use record counting instead (see §11.4).

Entry counting is provided by the base glossaries package and is enabled with \texttt{glsenableentrycount}. This keeps a count of the number of times an entry is marked as used, which is done by hooking into the unset and reset commands (see §5.10). The current running total can be obtained with \texttt{glsentrycurrcount}. The total from the end of the previous \LaTeX{} run can be obtained with \texttt{glsentryprevcount}.

Since entry counting relies on the first use flag, it doesn’t take the \texttt{glstext}-like commands into account.
6. Counting References

Entry counting is incompatible with \texttt{docdef=true}.

The glossaries-extra package modifies \texttt{\glsenableentrycount} to allow for the \texttt{entrycount} attribute. This means that you not only need to enable entry counting with \texttt{\glsenableentrycount}, but you also need to set the \texttt{entrycount} attribute (see below).

Prior to v1.49, the associated counter was reset back to 0 when the first use flag is reset. This behaviour is now only implemented if the following conditional is true:

\[
\texttt{\ifglsresetcurrcount \langle true \rangle \text{\else} \langle false \rangle \text{\fi}} \quad \text{initial: \texttt{\iffalse}}
\]

To (locally) change this conditional to true use:

\[
\texttt{\glsresetcurrcounttrue}
\]

To (locally) change this conditional to false use:

\[
\texttt{\glsresetcurrcountfalse}
\]

As from v1.49, the default is now false. Note that this conditional is also available with glossaries v4.50+.

Remember that entry counting only counts the number of times an entry is used by commands that change the first use flag. (That is, all those commands that mark the entry as having been used.) There are many commands that don’t modify this flag and they won’t contribute to the entry use count.

With just the base glossaries package, the associated entry counting commands, such as \texttt{\cgls}, are only available when entry counting has been activated with \texttt{\glsenableentrycount}. Whereas with glossaries-extra, those commands are always available but behave in the same way as the corresponding \texttt{\gls}-like commands if entry counting hasn’t been activated. The commands provided by the \texttt{shortcuts} options, such as \texttt{\ac} are defined to use \texttt{\cgls} instead of \texttt{\gls} etc so you can use them either with or without entry counting.

In order to activate entry counting with glossaries-extra, you not only need to use \texttt{\glsenableentrycount} but also need to specify the trigger value.

\[
\texttt{\GlsXtrEnableEntryCounting\{\langle category-list\rangle\}\{\langle trigger-value\rangle\}}
\]
6. Counting References

This command is provided as a shortcut to activate entry counting and assign the trigger value. This command performs the following:

- enables entry counting with \glsenableentrycount;
- redefines the \gls-like commands to do the equivalent \cgls commands (so you don’t need to keep track of which entries have entry counting enabled);
- sets the entrycount attribute to \langle trigger-value \rangle for all the supplied categories;
- disables the unit counting command (which is incompatible).

If you want to have different trigger values for different categories, you can set the entrycount attribute afterwards for the other category. For example:

```
\GlsXtrEnableEntryCounting{abbreviation, acronym}{1}
\glssetcategoryattribute{general}{2}
```

If you use \GlsXtrEnableEntryCounting multiple times, the repeated instances will simply set the entrycount attribute for the listed categories. So the above can also be written as:

```
\GlsXtrEnableEntryCounting{abbreviation, acronym}{1}
\GlsXtrEnableEntryCounting{general}{2}
```

The commands like \cgls behave like the corresponding \gls-like command if the entry count at the end of the previous run was more than a trigger value. With just the base glossaries package, this trigger value is 1. With glossaries-extra you can specify a different value.

```
\CGls[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩}  modifiers: * + ⟨alt-mod⟩
```

The appropriate trigger value must be set for the required category or categories.

As with the \gls-like commands, the \cgls set of commands may also be used with the star (*) or plus (+) modifiers or the modifier given by \GlsXtrSetAltModifier.

If the entry count at the end of the previous run doesn’t exceed the trigger value, the corresponding formatting command is used instead. For example, \cgls will use \cgls-format. The complete set of commands are:
6. Counting References

If the trigger value has been supplied for the entry’s category and is exceeded, this behaves like \gls otherwise it uses:

\cglsformat{⟨entry-label⟩}{⟨insert⟩}

This is redefined by glossaries-extra to test whether or not the entry has the regular attribute set or is an abbreviation:

\renewcommand*{\cglsformat}[2]{\glsifregular{#1}{\glsentryfirst{#1}}{% \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}#2}

This show the first use value if the entry is regular otherwise it will show the long form. The insert is appended at the end.

\cGlsp{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

If the trigger value has been supplied for the entry’s category and is exceeded, this behaves like \glsp otherwise it uses:

\cGlsformat{⟨entry-label⟩}{⟨insert⟩}

This is like \cglsformat but uses the plural commands.

\cGls[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

If the trigger value has been supplied for the entry’s category and is exceeded, this behaves like \Gls otherwise it uses:

\cGlsformat{⟨entry-label⟩}{⟨insert⟩}

This is like \cglsformat but uses the sentence case commands.

\cGlsp{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

If the trigger value has been supplied for the entry’s category and is exceeded, this behaves
6. Counting References

like \Glspl otherwise it uses:

\cGlsplformat{⟨entry-label⟩}{⟨insert⟩}

This is like \cGlsformat but uses the plural sentence case commands.

The glossaries-extra package provides some additional commands:

\cGLS[(options)]{(entry-label)}{(insert)}

modifiers: * + ⟨alt-mod⟩

If the trigger value has been supplied for the entry’s category and is exceeded, this behaves like \GLS otherwise it uses:

\cGLSformat{(entry-label)}{(insert)}

This simply uses \cGlsformat converted to uppercase.

\cGLSpl[(options)]{(entry-label)}{(insert)}

modifiers: * + ⟨alt-mod⟩

If the trigger value has been supplied for the entry’s category and is exceeded, this behaves like \GLSpl otherwise it uses:

\cGLSplformat{(entry-label)}{(insert)}

This simply uses \cGlsplformat converted to uppercase.

The test to determine whether or not an entry trips the trigger value is performed by:

\glsxtrifcounttrigger{(entry-label)}{(true)}{(false)}

This obtains the trigger value from the entry’s entrycount attribute.

Since these commands require information from the previous \LaTeX run, and extra \LaTeX call must be added to the build process (before the relevant indexing application).

For example, in the following document the trigger value is set to 1. The CSS entry is only used once (which doesn’t exceed the trigger). The HTML entry is used twice (which does exceed the trigger). The sample entry is only used once, but entry counting hasn’t been enabled on its category (the default general).
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\makeglossaries
\GlsXtrEnableEntryCounting{abbreviation}{1}
\newabbreviation{css}{CSS}{cascading style sheet}
\newabbreviation{html}{HTML}{hypertext markup language}
\newglossaryentry{sample}{name={sample},description={an example}}
\begin{document}
First use: \gls{css}, \gls{html} and \gls{sample}.
Next use: \gls{html}.
\printglossaries
\end{document}

If the document is saved in a file called myDoc.tex then the build process is:

\texttt{pdflatex myDoc}
\texttt{pdflatex myDoc}
\texttt{makeglossaries myDoc}
\texttt{pdflatex myDoc}

Note the second \LaTeX{} call before \texttt{makeglossaries}. The result is shown below:

Example 121: Entry counting according to category

First use: cascading style sheet, hypertext markup language (HTML) and sample. Next use: HTML.

\textbf{Glossary}

\begin{tabular}{ll}
\texttt{HTML} & hypertext markup language 1 \\
\texttt{sample} & an example 1
\end{tabular}

Note that the CSS entry only shows the long form, doesn’t appear in the glossary and doesn’t have a hyperlink. This is because the total count from the previous \LaTeX{} run doesn’t exceed the value (1, in this case) that triggers the normal behaviour of \gls{}. The HTML entry has a total count of 2 from the previous \LaTeX{} run, so it’s displayed as normal with the full form on first use and has a hyperlink to its entry in the glossary.

The sample entry is only used once, but it has the default general category, which doesn’t have the entrycount attribute set.
6. Counting References

Note that if the build process only had one \TeX call before running \texttt{makeglossaries}, the HTML entry would also not appear in the glossary. This is because on the first \TeX run, the total from the previous run is 0 (because there’s no information in the aux file).

The \texttt{glossaries-extra} package also provides the ability to count per sectional unit instead:

\begin{verbatim}
glsenableentryunitcount
\end{verbatim}

It’s not possible to enable both document-wide entry counting (\texttt{\glsenableentrycount}) and unit entry counting (\texttt{\glsenableentryunitcount}).

The unit entry counting provides separate totals for each section unit. As above, this uses the \texttt{entrycount} attribute to provide the trigger value but also requires the \texttt{unitcount} attribute, which should be set to the name of the appropriate counter, such as section or chapter.

Due to the asynchronous nature of \TeX’s output routine, discrepancies will occur in page spanning paragraphs if you use the page counter.

As before, there is a command provided to enable the feature and set the corresponding attributes at the same time:

\begin{verbatim}
\GlsXtrEnableEntryUnitCounting\{\langle category-list\rangle\}\{\langle trigger-value\rangle\}\{\langle counter\rangle\}
\end{verbatim}

This command performs the following:

\begin{itemize}
  \item enables unit entry counting with \texttt{\glsenableentryunitcount};
  \item redefines the \texttt{\gls} like commands to do the equivalent \texttt{\cgls} commands (so you don’t need to keep track of which entries have entry counting enabled);
  \item sets the \texttt{entrycount} attribute to the supplied trigger for all the supplied categories;
  \item sets the \texttt{unitcount} attribute to the supplied counter for all the supplied categories;
  \item disables the document-wide counting command (which is incompatible).
\end{itemize}

If you use \texttt{\GlsXtrEnableEntryUnitCounting} multiple times, the repeated instances will simply set the \texttt{entrycount} and \texttt{unitcount} attributes for the listed categories.

The counter value is used as part of a label, which means that \texttt{\the\langle counter-name\rangle} needs to be expandable. Since \texttt{hyperref} also has a similar requirement and provides \texttt{\the\langle counter-name\rangle} as an expandable alternative, \texttt{glossaries-extra} will use \texttt{\the\langle counter-name\rangle} if it exists otherwise it will use \texttt{\the\langle counter-name\rangle}.

316
The commands for accessing the totals, \texttt{\glsentrycurrcount} and \texttt{\glsentryprevcount} have different definitions with unit entry counting and will expand to the total for the current unit. The overall totals can be obtained with additional commands:

\texttt{\glsentryprevtotalcount\{\textit{entry-label}\}}

This expands to the overall total from the previous \LaTeX run.

\texttt{\glsentryprevmaxcount\{\textit{entry-label}\}}

This expands to the maximum per-unit total from the previous \LaTeX run.

For example:

\begin{verbatim}
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\GlsXtrEnableEntryUnitCounting{abbreviation}{2}{section}
\makeglossaries
\% category={abbreviation}:
\newabbreviation{html}{HTML}{hypertext markup language}
\newabbreviation{css}{CSS}{cascading style sheet}
\% category={general}:
\newglossaryentry{sample}{name={sample},description={sample}}
\begin{document}
\section{Sample}
Used once: \texttt{\gls{html}}.
Used three times: \texttt{\gls{css}} and \texttt{\gls{css}} and \texttt{\gls{css}}.
Used once: \texttt{\gls{sample}}.
\section{Another Sample}
Used once: \texttt{\gls{css}}.
\end{document}
\end{verbatim}
6. Counting References

\begin{itemize}
\item Used twice: \gls{html} and \gls{html}.
\item \texttt{\textbackslash printglossaries}
\item \texttt{\textbackslash end\{document\}}
\end{itemize}

As before, if the document is in a file called \texttt{myDoc.tex} then the build process is:

\begin{itemize}
\item \texttt{pdflatex myDoc}
\item \texttt{pdflatex myDoc}
\item \texttt{makeglossaries myDoc}
\item \texttt{pdflatex myDoc}
\end{itemize}

The result is:

Example 122: Entry unit counting (per section) according to category

1 Sample

Used once: hypertext markup language.
    Used three times: cascading style sheet (CSS) and CSS and CSS.
    Used once: sample.

2 Another Sample

Used once: cascading style sheet.
    Used twice: hypertext markup language and hypertext markup language.

Glossary

\begin{itemize}
\item CSS cascading style sheet 1
\item sample an example 1
\end{itemize}

In this document, the CSS entry is used three times in the first section. This is more than the trigger value of 2, so \gls{css} is expanded on first use with the short form used on subsequent use, and the CSS entries in that section are added to the glossary. In the second section, the CSS entry is only used once, which trips the suppression trigger, so in that section, the long form is used and \gls{css} doesn’t get a line added to the glossary file.

The HTML entry is used a total of three times, but the expansion and indexing suppression trigger is tripped in both sections because the per-unit total (1 for the first section and 2 for the second chapter) is less than or equal to the trigger value.
The sample entry has only been used once, but it doesn’t trip the indexing suppression because it’s in the \textit{general} category, which hasn’t been listed in \texttt{GlsXtrEnableEntryUnitCounting}.

The per-unit entry counting can be used for other purposes. In the following example document, the trigger value is set to zero, which means the index suppression won’t be triggered, but the unit entry count is used to automatically suppress the hyperlink for commands like \texttt{\textbackslash\gls} by modifying the \texttt{\glslinkcheckfirsthyperhook} which is used at the end of the macro the determines whether or not to suppress the hyperlink.

\begin{verbatim}
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\makeglossaries
\GlsXtrEnableEntryUnitCounting{general}{0}{page}
\newglossaryentry{sample}{name={sample},description={an example}}
\renewcommand*{\glslinkcheckfirsthyperhook}{\ifnum\glsentrycurrcount\glslabel>0 \setupglslink{hyper=false}\fi}
\begin{document}
A \texttt{\gls{sample}} entry. Next use: \texttt{\gls{sample}}.
\newpage
Next page: \texttt{\gls{sample}}. Again: \texttt{\gls{sample}}.
\printglossaries
\end{document}
\end{verbatim}

This only produces a hyperlink for the first instance of \texttt{\gls{sample}} on each page.
6. Counting References

Example 123: Enabling unit counting to hook into hyperlink setting


Glossary

sample an example 1, 2

The earlier warning about using the page counter still applies. If the first instance of \gls occurs at the top of the page within a paragraph that started on the previous page, then the count will continue from the previous page.

6.2. Link Counting

As from version 1.26, an alternative method of entry counting is to count the number of times the \gls-like or \glstext-like commands are used. (The “link” in this method’s name refers to the use of the internal command \@gls@link not to \hyperlink although \@gls@link may use \hyperlink when displaying the link text.)

To enable link counting use the preamble-only command:

\GlsXtrEnableLinkCounting[⟨parent counter⟩]{⟨categories⟩}

where ⟨categories⟩ is a list of category labels. The optional argument ⟨parent counter⟩ may be used to identify a parent counter (which must already be defined). If present, the associated link counter will be reset when the parent counter is incremented. This command automatically sets the linkcount attribute for the given categories. If the optional argument is present, it also sets the linkcountmaster attribute.

When enabled, the \gls-like and \glstext-like commands will increment the associated counter using

\glsxtrinclinkcounter{⟨counter⟩}

This just does \stepcounter{⟨counter⟩} by default but if you need \refstepcounter instead, just redefine this command:

\renewcommand*{\glsxtrinclinkcounter}[1]{{\refstepcounter{#1}}

320
You can access the internal count register using:

\GlsXtrLinkCounterValue{⟨entry-label⟩}

where ⟨label⟩ is the entry’s label. This will expand to 0 if the register hasn’t been defined.

It’s also possible to access the display value (\the⟨counter⟩) using

\GlsXtrTheLinkCounter{⟨entry-label⟩}

(This will expand to 0 if the counter hasn’t been defined.)

In order to conserve resources, the counter is only defined when it first needs to be incremented so terms that have been defined but haven’t been used in the document won’t have the associated count register allocated.

You can test if the counter has been defined using:

\GlsXtrIfLinkCounterDef{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}

This expands to ⟨true⟩ if the link counter associated with the entry identified by ⟨entry-label⟩ has been defined, otherwise expands to ⟨false⟩.

The counter name can be obtained using

\GlsXtrLinkCounterName{⟨entry-label⟩}

This simply expands to the counter name associated with the entry given by ⟨entry-label⟩ without any check for existence. For example, to change the display command (\the⟨counter⟩) using etoolbox:

\csdef{\the\GlsXtrLinkCounterName{duck}}\%\
{\Roman{\GlsXtrLinkCounterName{duck}}}\

This is useful if you just want to change the display for specific entries but isn’t convenient if you want to change the display for all entries. Instead, it’s simpler to redefine \GlsXtrTheLinkCounter. For example:
In both cases, the redefinition should be implemented after \GlsXtrEnableLinkCounting.

Here’s an example document that uses link counting to disable the hyperlink after the first reference. This redefines \glslinkpresetkeys (which is used by both \gls and \glstext) instead of \glslinkcheckfirsthyperhook (which is used by \gls but not by \glstext).

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage{glossaries-extra}
\makeglossaries
\renewcommand*{\glslinkpresetkeys}{\ifnum\GlsXtrLinkCounterValue{\glslabel}>1
\setupglslink{hyper=false}\fi}
\GlsXtrEnableLinkCounting{general}
\newglossaryentry{sample1}{name={sample1},description={an example}}
\newglossaryentry{sample2}{name={sample2},description={another example}}
\newabbreviation{ex}{ex}{example}
\begin{document}
\section{Sample Section}
\Gls{sample1}, \gls{sample2} and \gls{ex}.
\Glstext{sample1} and \gls{ex} again.
\section{Another Sample Section}
\end{document}
6. Counting References

\Gls{sample1}, \gls{sample2} and \gls{ex}.

\printglossaries
\end{document}

The use of \glslinkpresetkeys means that the options can override this. For example

\gls[hyper=true]{sample1}

(or simply \gls+{sample1}) will override the hyper=false setting in \glslinkpresetkeys. If \glslinkpostsetkeys is used instead, the hyper=false setting will override the setting provided in the optional argument. The resulting document is shown below:

Example 124: Link counting used to selectively suppress hyperlinks

1 Sample Section

Sample1, sample2 and example (ex). Sample1 and ex again.

2 Another Sample Section

Sample1, sample2 and ex.

Glossary

ex example 1

sample1 an example 1

sample2 another example 1

The abbreviation category doesn’t have the linkcount attribute set (since it’s not listed in the argument of \GlsXtrEnableLinkCounting). This means that \GlsXtrLinkCounterValue always expands to 0 for the abbreviation (ex), so the inequality test:

\ifnum\GlsXtrLinkCounterValue{\glslabel}>1

will always be false. This means that the abbreviation won’t have hyper=false applied. If the test is changed to
6. Counting References

\ifnum\GlsXtrLinkCounterValue{\gllabel}=1
\else
  \setupglslink{hyper=false}\
\fi

Then the abbreviation will always have \texttt{hyper=false} applied.
To reset the counter every section use the optional argument to set the parent counter:

\GlsXtrEnableLinkCounting[\texttt{section}]{\texttt{general}}
7. Multi (or Compound) Entries

Nested entries (where the entry definition references other entries) are discussed in §5.4. This chapter deals with occasions where a term or phrase may consist of multiple sub-terms that are independently defined. (Examples in §7.1.5 and §7.1.6 provide workarounds for nested entries.)

For example, the names of bacteria, such as *Clostridium botulinum* and *Clostridium perfringens*, are made up of the genus (for example, *Clostridium*) and the species (for example, *botulinum* or *perfringens*). The genus is often abbreviated after first use. For example, *C. botulinum*. However, if the name is defined as a single term consisting of both the genus and species then it’s not possible to apply the abbreviation when a different species with the same genus is used. Consider the following document:

```latex
\documentclass{article}
\usepackage{glossaries-extra}
\setabbreviationstyle{long-only-short-only}
\newabbreviation{cbot}{C. botulinum}{Clostridium botulinum}
\newabbreviation{cperf}{C. perfringens}{Clostridium perfringens}
\begin{document}
\gls{cbot}, \gls{cbot}, \gls{cperf}.
\end{document}
```

The result is:

Clostridium botulinum, C. botulinum, Clostridium perfringens.

However, it should more typically be:

Clostridium botulinum, C. botulinum, C. perfringens.

In this case, the genus should actually be a separate definition:

```latex
\documentclass{article}
\usepackage{glossaries-extra}
```
7. Multi (or Compound) Entries

This is quite awkward to write. This chapter describes how to provide a shortcut for this kind of construct. Each term should be defined as normal (as in the above example), and a “multi-entry” label is then defined with the list of labels of the entries that need to be referenced.

\multiglossaryentry\[\langle options\rangle\]{\langle multi-label\rangle}[\langle main-label\rangle]{\langle entry-label-list\rangle}  

This defines a multi-entry set with the label \langle multi-label\rangle, consisting of the entries whose labels are listed in \langle entry-label-list\rangle, where the main entry (which must be present in \langle entry-label-list\rangle) is identified by \langle main-label\rangle. If \langle main-label\rangle is omitted, it’s assumed to be the final label in \langle entry-label-list\rangle. The main entry is described in more detail in §7.2.

The entries in \langle entry-label-list\rangle must already be defined (using commands like \texttt{\newglossaryentry} or \texttt{\newabbreviation}).

The \langle options\rangle are a comma-separated list of options to override the current settings and are described in §7.9.

The earlier example can now be modified to include the following:

\multiglossaryentry\{cbot\}\{clostridium,botulinum\}
\multiglossaryentry\{cperf\}\{clostridium,perfringens\}

These commands must come after the \texttt{clostridium}, \texttt{botulinum} and \texttt{perfringens} definitions.

Once defined, a multi-entry set can be referenced in the document using commands like:

\mgls\[\langle options\rangle\]{\langle multi-label\rangle}[\langle insert\rangle]  

This command essentially does \texttt{\gls\{label\}} for each item in the \langle label list\rangle (with separators, see §7.4). If the final optional argument \langle insert\rangle is provided, it will be applied to the
7. Multi (or Compound) Entries

final (non-skipped) element in the list. So the document body in the above example, can be rewritten as:

\texttt{\gls{cbot}, \gls{cbot}, \gls{cperf}.

There are some variants of \texttt{\gls} listed in §7.11. The available \texttt{\langle options\rangle} are listed in §7.10. They are applied after the \texttt{\multiglossaryentry} options and will override settings for the individual entries.

You can't use \texttt{\langle multi-label\rangle} in commands like \texttt{\gls} as this label represents a set of entry labels not a single entry.

The \texttt{\multiglossaryentry} command will generate an error if the label has already been defined as a multi-entry.

This does nothing if a multi-entry set with the given label has already been defined otherwise it will act like \texttt{\multiglossaryentry}. Notes and associated commands applying to \texttt{\multiglossaryentry} also apply to \texttt{\providemultiglossaryentry} unless otherwise stated.

\texttt{\multiglossaryentry} may be placed anywhere after the entries listed in \texttt{\langle label list\rangle} have been defined. A multi-entry label can't be referenced (with commands like \texttt{\gls}) before it has been defined.

There is limited support for \texttt{docdef=true}. The multi-entry definition can be picked up from the aux file on the next run to allow cross-references in any glossaries that occur at the start of the document. Any changes made with commands like \texttt{\glsSetMain} won't be carried over to the next run.

By default \texttt{\multiglossaryentry} will be localised to the current scope. If you want to globally define a multi-entry you need to first switch on global definitions with:

\texttt{\multiglossaryentryglobaltrue}
7. Multi (or Compound) Entries

To switch back to local definitions use:

\multiglossaryentryglobalfalse

You can test if this setting is on with:

\ifmultiglossaryentryglobal \true\ else \false\ fi \ initial: \iffalse

If you want to change the multi-entry options (locally) you can use:

\mglsSetOptions{⟨multi-label⟩}{⟨new-options⟩}

This removes the original options and replaces them with ⟨new-options⟩. If you want to (locally) append to the existing options, use:

\mglsAddOptions{⟨multi-label⟩}{⟨new-options⟩}

Note that \multiglossaryentry doesn’t make any adjustments to the component entries. You will need to use the parent key when you define the entries if you want a hierarchical structure in your glossary. (See the example in §7.1.1.)

If you don’t want the other elements in the glossary, you can suppress the indexing with indexothers=false (§7.9.1) or put them in an ignored glossary. For example:

\newignoredglossary{common}
\newabbreviation[type={common}]{clostridium}{C.}{Clostridium}

The ⟨multi-label⟩ can’t be used in commands like \gls since the label refers to a set of entry labels not to an individual entry. Similarly, an individual entry label can’t be used in commands like \mgls. It is possible (although potentially confusing) to use the same label for a multi-entry as for an individual entry (see the example in §7.1.6). Context will determine which is meant, except in the case of the cross-referencing fields (see, seealso and alias) where the cross-referenced label will first be tested if it’s a known multi-entry label.

If you don’t want to have to keep track of which labels refer to multi-entries and which refer to individual entries you can use:

\GlsXtrMglsOrGls{⟨mgls cs⟩}{⟨gls cs⟩}{⟨modifier⟩}{⟨options⟩}{⟨label⟩}{⟨insert⟩}
where \(<mgls\ cs>\) is the \mgls-like command to use if \(<label>\) has been defined as a multi-entry and \(<gls\ cs>\) is the \gls-like or \glstext-like command to use otherwise. The \(<modifier>\) may be omitted, otherwise it’s the modifier that may be used with \mgls or \gls (asterisk *, plus + or the token identified with \GlsXtrSetAltModifier). The modifier and remaining options are passed to the relevant command (\(<mgls\ cs>\) or \(<gls\ cs>\)).

You may prefer to define your own shortcut commands for common combinations. For example, (assuming these commands haven’t already been defined by the shortcuts option):

\begin{verbatim}
\newcommand{\ac}{\GlsXtrMglsOrGls{\mgls}{\gls}}
\newcommand{\acp}{\GlsXtrMglsOrGls{\mglsmainpl}{\glspl}}
\newcommand{\Ac}{\GlsXtrMglsOrGls{\Mgls}{\Gls}}
\newcommand{\Acp}{\GlsXtrMglsOrGls{\Mglsmainpl}{\Glspl}}
\end{verbatim}

7.1. Examples

7.1.1. Example: Hierarchical

Bacteria names are represented by the genus (for example, Clostridium) followed by the species (for example, botulinum). This example has the genus as a parent of the species.

\begin{verbatim}
\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[stylemods=bookindex,style=bookindex]{glossaries-extra}
\makeglossaries
\newcommand{\latinname}[1]{\emph{#1}}
\glssetcategoriesattributes
{genus,species}{textformat,glossnamefont}{latinname}
\setabbreviationstyle[genus]{long-only-short-only-desc}
\newabbreviation
{category={genus},description={}}
{clostridium}{C.}{Clostridium}
\newglossaryentry{botulinum}{name={botulinum},category={species},
description={},parent={clostridium}}
\newglossaryentry{perfringens}{name={perfringens},
\end{verbatim}
7. Multi (or Compound) Entries

\begin{verbatim}
category={species},description={},parent={clostridium}
\newglossaryentry{tetani}{name={tetani},category={species},
description={},parent={clostridium}}
\multiglossaryentry{cbot}{clostridium,botulinum}
\multiglossaryentry{cperf}{clostridium,perfringens}
\multiglossaryentry{ctet}{clostridium,tetani}
\multiglossaryentrysetup{indexothers=false,hyper=allmain}
\begin{document}
First use: \mgls{cbot}, \mgls{cperf}, \mgls{ctet}.
Next use: \mgls{cbot}, \mgls{cperf}, \mgls{ctet}.
\printglossaries
\end{document}
\end{verbatim}

This suppresses the indexing of the non-main elements (in this case, the genus). However
the genus is included in the glossary (without a location list) because it’s the parent of the
species (which are indexed).

Example 125: Multi-entries: hierarchical

First use: \textit{Clostridium botulinum, C. perfringens, C. tetani.}
Next use: \textit{C. botulinum, C. perfringens, C. tetani.}

\textbf{Glossary}

\begin{itemize}
  \item \textit{Clostridium botulinum 1}
  \item \textit{perfringens 1}
  \item \textit{tetani 1}
\end{itemize}

The \texttt{hyper=allmain} option makes the entire content of each \texttt{\mgls} a hyperlink to the
main entry in the glossary.

7.1.2. Example: Suffix

This example has a minor modification to the previous one. In this case the multi-entries are
defined with a suffix:
7. Multi (or Compound) Entries

The rest of the document is as in §7.1.1.

Example 126: Multi-entries: hierarchical with first-use suffix

First use: *Clostridium botulinum* (botulism), *C. perfringens* (gas gangrene), *C. tetani* (tetanus).
Next use: *C. botulinum*, *C. perfringens*, *C. tetani*.

**Glossary**

\begin{center}
\begin{tabular}{ll}
\textbf{C} & \textit{perfringens 1} \\
\textit{Clostridium} & \textit{tetani 1} \\
\textit{botulinum 1} & \\
\end{tabular}
\end{center}

### 7.1.3. Example: Category Suffix

This is an alternative to the previous example. Instead of storing the extra information in the `firstsuffix` key, the information is stored in the `user1` key of the last element (the species). A category suffix is used to look up the field and append it.

\begin{verbatim}
\newglossaryentry{botulinum}{name={botulinum},category={species},
user1={botulism},
description={},parent={clostridium}}
\newglossaryentry{perfringens}{name={perfringens},category={species},
user1={gas gangrene},
description={},parent={clostridium}}
\newglossaryentry{tetani}{name={tetani},category={species},
user1={tetanus},
description={},parent={clostridium}}
\mglsdefcategorysuffix{bacteria}{}
\end{verbatim}
7. Multi (or Compound) Entries

Example 127: Multi-entries: hierarchical with category suffix

First use: *Clostridium botulinum* (botulism), *C. perfringens* (gas gangrene), *C. tetani* (tetanus).
Next use: *C. botulinum*, *C. perfringens*, *C. tetani*.

Glossary

<table>
<thead>
<tr>
<th>C</th>
<th>perfringens 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clostridium</td>
<td>tetani 1</td>
</tr>
<tr>
<td>botulinum</td>
<td></td>
</tr>
</tbody>
</table>

7.1.4. Example: Separators

The first example (§7.1.1) can be modified so that the species are also abbreviations. In this case, the separators are modified to suppress the space (\relax) if both the genus and species are abbreviated, or to use a non-breaking space (\textasciiplain{~}) between the genus short form (shown on subsequent use) and the species long form (shown on first use). If the genus is showing the long form (first use) then a normal space is used.

Note that the separator attributes apply to the category of the element before the separator (not to the multi-entry category).
7. Multi (or Compound) Entries

```
\setabbreviationstyle[species]{long-only-short-only-desc}
\newabbreviation[category={species},
                   description={},parent={clostridium}]{botulinum}{bot.}{botulinum}
\newabbreviation[category={species},
                   description={},parent={clostridium}]{perfringens}{per.}{perfringens}
\newabbreviation[category={species},
                   description={},parent={clostridium}]{tetani}{tet.}{tetani}
```

This will cause a double dot at the end of the second sentence, which can be suppressed using the \texttt{discardperiod} and \texttt{retainfirstuseperiod} attributes.

```
\glssetcategoriesattributes{species}{discardperiod,retainfirstuseperiod}{true}
```

This works because the final element’s post-link hook is transferred to the multi-entry post-link hook, which can detect the sentence terminating period. If the post-link hook settings are changed, for example, to \texttt{postlinks=all, mpostlink=false} then the feature won’t work as the final element’s post-link hook can’t detect the period (because \texttt{\gls} is embedded too deeply inside the internal workings of \texttt{\mgls}).

\begin{example}{Multi-entries: separators}
\begin{itemize}
\item First use: \textit{Clostridium botulinum, C. perfringens, C. tetani.}
\item Next use: \textit{C.bot., C.per., C.tet.}
\end{itemize}
\end{example}

\begin{glossary}
\begin{tabular}{ll}
\textit{C} & perfringens 1 \\
\textit{Clostridium} & tetani 1 \\
\textit{botulinum} & 1
\end{tabular}
\end{glossary}

7.1.5. Example: Skipping Elements (Fragment Element)

This example is an alternative way of dealing with nested links (see §5.4).
7. Multi (or Compound) Entries

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[stylemods,style=long]{glossaries-extra}
\makeglossaries
\setabbreviationstyle{long-short-sc}
\newabbreviation{ssi}{ssi}{server-side includes}
\newabbreviation{html}{html}{hypertext markup language}
\setabbreviationstyle[combinedabbrv]{long-only-short-sc-only}
\newabbreviation[category={combinedabbrv},
description={\glsxtrshort{ssi} enabled \glsxtrshort{html}}]{shtml-frag}{shtml}{enabled}
\glssetcategoryattribute{combinedabbrv}{multioptions}
{usedskipothers=,firstsuffix=\glsxtrshort{html}}
\multiglossaryentry[category=combinedabbrv]
{shtml}[shtml-frag]{ssi,shtml-frag,html}
\begin{document}
Individual elements first use: \gls{ssi} and \gls{html}.

Individual elements next use: \gls{ssi} and \gls{html}.

Multi-entry first use: \mgl{shtml}.

Multi-entry next use: \mgl{shtml}.

Resetting all\glsresetall\mglreset{shtml}:

Multi-entry first use: \mgl{shtml}.

Multi-entry next use: \mgl{shtml}.

Individual elements: \gls{ssi} and \gls{html}.
\printglossaries
This uses the \texttt{multioptions} attribute to skip “other” elements on subsequent use. The problematic abbreviation (SHTML) is defined as a fragment that simply expands to “enabled” on first use. Note that the description has to be supplied for the glossary. The resulting document is shown below.

Example 129: Multi-entries: skipping elements

Individual elements first use: \texttt{server-side includes (SSI)} and hypertext markup language (HTML).

Individual elements next use: SSI and HTML.

Multi-entry first use: \texttt{SSI enabled HTML (SHTML)}.

Multi-entry next use: \texttt{SHTML}.

Resetting all:

Multi-entry first use: server-side includes (SSI) enabled hypertext markup language (HTML) (SHTML).

Multi-entry next use: \texttt{SHTML}.

Individual elements: SSI and HTML.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
HTML & hypertext markup language 1 \\
\hline
SHTML & SSI enabled HTML 1 \\
SSI & server-side includes 1 \\
\hline
\end{tabular}
\caption{Glossary}
\end{table}

The key difference here from the example using \texttt{\textbackslash glsp\textbackslash s} at the end of §5.4 is that the individual elements hyperlink to their respective entries in the glossary on first use of \texttt{\textbackslash g\textbackslash l\textbackslash s\textbackslash s}.

The problem is that with the \texttt{colorlinks} package option, it’s not obvious where the hyperlinks start and end. The suffix (SHTML) will hyperlink to the “shtml” entry in the glossary, so the “enabled” hyperlink is redundant. The simplest fix for this is to add \texttt{hyper=notmainfirst} to the option list, which will prevent “enabled” from being a hyperlink.

Another problem occurs where \texttt{\textbackslash g\textbackslash l\textbackslash s} is used before the individual elements are used, which leads to their full expansion with a confusing amount of parentheses. A simple solution is to use the option \texttt{mglsopts=unsetothers}, which will unset the other (not-main) elements first. This can be localised with \texttt{\textbackslash presetlocal} but \texttt{\textbackslash g\textbackslash l\textbackslash s} will then unset the first use flag globally, which means that the other elements won’t show the full form when they are first used on their own after \texttt{\textbackslash g\textbackslash l\textbackslash s}. This can be switched to a local unset with \texttt{\textbackslash others/local}.

The complete set of options are now:
7. Multi (or Compound) Entries

Example 130: Multi-entries: skipping elements (unsettting others)

Individual elements first use: server-side includes (SSI) and hypertext markup language (HTML).
Individual elements next use: SSI and HTML.
Multi-entry first use: firstsuffix, SSI enabled HTML.
Multi-entry next use: firstsuffix, SSI SHTML HTML.
Resetting all:
Multi-entry first use: firstsuffix, SSI enabled HTML.
Multi-entry next use: firstsuffix, SSI SHTML HTML.
Individual elements: server-side includes (SSI) and hypertext markup language (HTML).

Glossary

<table>
<thead>
<tr>
<th>HTML</th>
<th>hypertext markup language 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHTML</td>
<td>SSI enabled HTML 1</td>
</tr>
<tr>
<td>SSI</td>
<td>server-side includes 1</td>
</tr>
</tbody>
</table>

This method still has two main drawbacks: the description must be added manually and the long form can’t be accessed with \glsxtrlong. The next example provides an alternative approach.

7.1.6. Example: Skipping Elements (Prefix and Post-Link Hooks)

This is a modified version of the previous example. In this case, the main element isn’t a fragment and also happens to have the same label as the multi-entry set. (\gls{shtml} references the multi-entry label and \gls{shtml} references the individual entry.)

In this case, the nested parts are marked up with custom commands:
This means that it’s no longer necessary to manually insert the description and the long form can be accessed as usual with  \texttt{\glsxtrshort{shtml}}. Note that it is necessary to define the custom commands robustly otherwise they will need to be protected against premature expansion:

\begin{verbatim}
\newcommand{\combinedpre}[1]{\glsps{#1}}
\newcommand{\combinedpost}[1]{\glsps{#1}}
\newabbreviation{shtml}{shtml}
{{}\protect\combinedpre{ssi} enabled \protect\combinedpost{html}}
\end{verbatim}

In both cases, an initial empty group is added to guard against any sentence case commands, such as \texttt{\Glsxtrlong}.

The abbreviations all use the \texttt{long-postshort-sc-user} style, which places the short form in the post-link hook on first use. The \texttt{\gls} post-link hook for the main element can be transferred to the \texttt{\mgls} post-link using:

\begin{verbatim}
\mpostlinkelement=main
\end{verbatim}

All elements have their individual post-link hooks suppressed by default. As in the previous example, the other elements can be skipped on subsequent use:

\begin{verbatim}
\usedskipothers
\end{verbatim}

Within \texttt{\mgls}, the nested content needs to be suppressed, which can be done by redefining the custom commands. This can be done in the multi-entry prefix. Since the entire content of \texttt{\mgls} (except for the final multi-entry post-link hook) occurs inside a group, this redefinition will be localised.

The complete document is as follows:
7. Multi (or Compound) Entries

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[stylemods,style=long]{glossaries-extra}
\makeglossaries
\setabbreviationstyle{long-postshort-sc-user}
\newabbreviation{ssi}{ssi}{server-side includes}
\newabbreviation{html}{html}{hypertext markup language}
\newrobustcmd{\combinedpre}[1]{\gls{#1}}
\newrobustcmd{\combinedpost}[1]{\gls{#1}}
\newabbreviation{shtml}{shtml}
{}\combinedpre{ssi} enabled \combinedpost{html}
\glssetcategoryattribute{combinedabbrv}{multioptions}
\% 
\% mpostlinkelement=main, 
\% usedskipothers 
\}%
\multiglossaryentry
\% [category=combinedabbrv] 
\{shtml\}[shtml]{ssi,shtml,html}
\mglsetcategoryprefix{combinedabbrv}{\% 
\renewcommand{\combinedpre}[1]{\ignorespaces}\% 
\renewcommand{\combinedpost}[1]{\unskip}\% 
}\% 
\begin{document}
Individual elements first use: \gls{ssi} and \gls{html}.

Individual elements next use: \gls{ssi} and \gls{html}.

Multi-entry first use: \mgls{shtml}.

Multi-entry next use: \mgls{shtml}.

338
7. Multi (or Compound) Entries

Individual entry first use: \gls{shtml}.

Resetting all\glsresetall\mglsresetall:

Multi-entry first use: \mgls{shtml}.

Multi-entry next use: \mgls{shtml}.

Individual elements: \gls{ssi} and \gls{html}.

Resetting all\glsresetall\mglsresetall:

Individual entry first use: \gls{shtml}.

Multi-entry first use: \mgls{shtml}. (Wrong!)

\printglossaries
\end{document}

This now produces:
7. *Multi (or Compound) Entries*

Example 131: Multi-entries: skipping elements (prefix and post-link hooks)

- Individual elements first use: server-side includes (SSI) and hypertext markup language (HTML).
- Individual elements next use: SSI and HTML.
- Multi-entry first use: SSI enabled HTML (SHTML).
- Multi-entry next use: SHTML.
- Individual entry first use: SHTML.
- Resetting all:
- Multi-entry first use: server-side includes enabled hypertext markup language (SHTML).
- Multi-entry next use: SHTML.
- Individual elements: SSI and HTML.
- Resetting all:
- Individual entry first use: SSI enabled HTML (SHTML).
- Multi-entry first use: server-side includes SHTML hypertext markup language. (Wrong!)

**Glossary**

- **HTML**  hypertext markup language 1
- **SHTML**  SSI enabled HTML 1
- **SSI**  server-side includes 1

Note the last two paragraphs, which highlights what happens if \gls{shtml} is used before \mgl{shtml} when neither of the other elements (ssi and html) have been used. The final instance of \gls has produced the wrong result. This is because it’s the first use of the multi-entry shtml but not the first use of the individual entry shtml.

One way around this is to modify the prefix to ensure that the main element’s first use flag matches the multi-entry’s first use flag:

\mgl{combinedabbrv}{%  
\renewcommand{\combinedpre}[1]{\ignorespaces}%  
\renewcommand{\combinedpost}[1]{\unskip}%  
\mglisfirstuse  
{%\glslocalreset{\mglcurrentmainlabel}}%  
{%\glslocalunset{\mglcurrentmainlabel}}%
}
7. Multi (or Compound) Entries

The showtargets=annoteleft option can be used to mark up the links with the targets. For example, the first instance of \mgl{shtml} will show as:

Multi-entry first use: [glo:si].SSI. [glo:shtml].enabled. [glo:html].HTML. 

Each entry has an individual hyperlink to its own glossary item, which may be confusing. This can be made clearer by suppressing the main element link on first use with:

\[\text{hyper=notmainfirst}\]

(as in the previous example), and adjusting the abbreviation style so that the parenthetical content in the post-link hook has a hyperlink:

\[\text{\renewcommand*{\glsxtruserparen}[2]{% \glsxtrfullsep{#2}% \glsxtrparen {% \glshyperlink{[#1]{#2}% \ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{}}% % }% }}\]

The remaining problem is how to deal with the possibility that \mgl{shtml} may come before the first use of the other elements. For example:

Multi-entry first use: \mgl{shtml}. 

Individual elements: \glsssi and \glsshtml.

This leads to:

Multi-entry first use: server-side includes enabled hypertext markup language (shtml). 
Individual elements: SSI and HTML.

This means that the abbreviations SSI and HTML aren’t explained in the document text. One way around this is to only locally unset the other element first use flags:

341
7. Multi (or Compound) Entries

\glssetcategoryattribute{combinedabbrv}{multioptions}
{%
 hyper=notmainfirst,
 mpostlinkelement=main,
 usedskipothers=,
 mglsopts=others=local
%
}

With the above setting, the following:

\glsresetall \mglsresetall

Multi-entry first use: \gls{shtml}.

Multi-entry next use: \gls{shtml}.

Individual elements: \gls{ssi} and \gls{html}.

will now produce:

Multi-entry first use: server-side includes enabled hypertext markup language (shtml).

Multi-entry next use: shtml.

Individual elements: server-side includes (ssi) and hypertext markup language (HTML).

7.2. Main and Other Elements

The list of labels provided in the final argument of \multiglossaryentry consists of a main element and all the other elements. If the main element isn’t identified in the optional argument, it’s assumed to be the final element in the list.

The main element allows you to determine which target to use if you want the entire content of \gls to be a single hyperlink. You can also use the settings described in §7.9 to only index the main element.

You can change the main element using: \glsSetMain The new main label provided in the second argument must be in the list corresponding to ⟨multi-label⟩. This change is locally applied to the current scope. Note that if you are using bib2gls, this change in the document can’t be detected.

The main element can also be used to identify which element should be displayed in the plural with \glsmainpl. For example:
In the above, two multi-entries are defined: greatgrebe and littlegrebe. In both cases the main element is grebe (the last element). Using \mglsp will show the plural for all elements, but using \mglsmainpl will only use the plural for the main element (grebe). For example:

\begin{verbatim}
\mglsp{greatgrebe}, \mglsp{littlegrebe}.
\mglsmainpl{greatgrebe}, \mglsmainpl{littlegrebe}.
\end{verbatim}

produces:

\begin{verbatim}
\end{verbatim}

7.3. Prefixes and Suffixes

A multi-entry may have associated prefixes and suffixes. These are scoped and are placed outside of the hyperlinks and encapsulating commands. They are not affected by case-changing commands, such as \Mgls. If you want a prefix to obey case-changing, use the \mpgls-like commands instead (§7.11.4).

The prefix is inserted with:

\begin{verbatim}
\mglsprefix
\end{verbatim}

The default definition is:
7. Multi (or Compound) Entries

\newcommand*{\mglsprefix}{%
  \ifdefempty\mglscurrentcategory
  {\mglscurrentprefix}%
  {%
    \mglshascategoryprefix{\mglscurrentcategory}%
    {\mglsusecategoryprefix{\mglscurrentcategory}}%
    {\mglscurrentprefix}%
  }%
}\}

This will insert the current prefix unless there is prefix command associated with the current category.

The suffix is inserted with:

\mglssuffix

This command is defined as follows:

\newcommand*{\mglssuffix}{%
  \ifdefempty\mglscurrentcategory
    {\ifdefempty\mglscurrentsuffix{}{\space(\mglscurrentsuffix)}}%
  {%
    \mglshascategorysuffix{\mglscurrentcategory}%
    {\mglsusecategorysuffix{\mglscurrentcategory}}%
    {\ifdefempty{\mglscurrentsuffix}{}{\space(\mglscurrentsuffix)}}%
  }%
}\}

If there is a suffix associated with the current category, that will be used, otherwise if the current suffix isn’t empty this inserts a space followed by the current suffix in parentheses. You can access the label of the last (non-skipped) element with \mglslastelementlabel.

Note that in both cases the category corresponds to the multi-entry category (see §7.8).

To define a category-dependent prefix, use:

\mglsdefcategoryprefix{\langle category-label\rangle}{\langle definition\rangle}

You can reference the current prefix with \mglscurrentprefix within \langle definition\rangle.

To define a category-dependent suffix, use:

\mglsdefcategorysuffix{\langle category-label\rangle}{\langle definition\rangle}
7. Multi (or Compound) Entries

You can reference the current suffix with \mglscurrentsuffix within \textit{definition}.

The default definition of \mglsprefix tests if there is a category prefix using:

\mglshascategoryprefix{\langle category-label \rangle}{\langle true \rangle}{\langle false \rangle}

This does \textit{true} if a prefix has been assigned to the given category, otherwise it does \textit{false}.

If you need to obtain the prefix for a particular category, you can use:

\mglssusecategoryprefix{\langle category-label \rangle}

This expands to the prefix, if set, for the given category or to nothing otherwise.

The default definition of \mglssuffix tests if there is a category suffix using:

\mglshascategoriesuffix{\langle category-label \rangle}{\langle true \rangle}{\langle false \rangle}

This does \textit{true} if a suffix has been assigned to the given category, otherwise it does \textit{false}.

If you need to obtain the suffix for a particular category, you can use:

\mglssusecategoriesuffix{\langle category-label \rangle}

This expands to the suffix, if set, for the given category or to nothing otherwise.

The current prefix \mglscurrentprefix and \mglscurrentsuffix are obtained as follows:

- if this is the first use of the multi-entry (§7.7) then the \textit{prefix} is set to the value of the \texttt{firstprefix} option and the \textit{suffix} is set to the value of the \texttt{firstsuffix} option;
- otherwise the \textit{prefix} is set to the value of the \texttt{usedprefix} option and the \textit{suffix} is set to the value of the \texttt{usedsuffix} option.

The prefix and suffix (if set) are placed outside of the hyperlink and text formatting encapsulator. They are not affected by case-changing commands such as \texttt{MglS} or \texttt{MGLS}.

For example:

\setabbreviationstyle{long-only-short-only}
\newabbreviation{clostridium}{C.}{Clostridium}
\newglossaryentry{botulinum}{name={botulinum},description={}}
7. Multi (or Compound) Entries

\multiglossaryentry[firstsuffix=botulism] {cbot}\{clostridium, botulinum\}

On first use, this produces (assuming the “clostridum” element hasn’t been used previously):

Clostridium botulinum (botulism).

On subsequent use, this produces:

C. botulinum.

7.4. Separators

The separators between each instance of \gls are given by the following commands, which all take two arguments. The first argument is the label of the previous element. The second argument is the label of the following element.

\glscombinedsep\{⟨prev label⟩\}{⟨next label⟩}

This is inserted between two entries that have both been marked as used. The default definition is:

\newcommand*{\glscombinedsep}[2]{\% \\glshasattribute{#1}{combinedsep}\% \\glsgetattribute{#1}{combinedsep}\% \{ \} }

This will use the combinedsep attribute for the ⟨prev label⟩’s category, if set. Otherwise this just does a space. Note that this ignores the second argument.

\glscombinedfirstsep\{⟨prev label⟩\}{⟨next label⟩}

This is inserted between two entries where only the next entry has been marked as used. The default definition is:
7. Multi (or Compound) Entries

\newcommand*{\glscombinedfirstsep}[2]{%
  \glshasattribute{#1}{combinedfirstsep}\%
  \glsgetattribute{#1}{combinedfirstsep}\%
  \glscombinedsep{#1}{#2}\%
}\newcommand*{\glscombinedfirstsepprev}{⟨prev label⟩}{⟨next label⟩}

This will use the combinedfirstsep attribute for ⟨prev label⟩‘s category, if set. If that attribute isn’t set, \glscombinedsep is used.

\newcommand*{\glscombinedsepfirst}[2]{%
  \glshasattribute{#1}{combinedsepfirst}\%
  \glsgetattribute{#1}{combinedsepfirst}\%
  \glscombinedsep{#1}{#2}\%
}\newcommand*{\glscombinedsepfirstprev}{⟨prev label⟩}{⟨next label⟩}

This is inserted between two entries where only the previous entry has been marked as used. The default definition is:

\newcommand*{\glscombinedfirstsepfirst}[2]{%
  \glshasattribute{#1}{combinedfirstsepfirst}\%
  \glsgetattribute{#1}{combinedfirstsepfirst}\%
  \glscombinedsep{#1}{#2}\%
}\newcommand*{\glscombinedfirstsepprevprev}{⟨prev label⟩}{⟨next label⟩}

This will use the combinedfirstsepfirst attribute for ⟨prev label⟩‘s category, if set. If that attribute isn’t set, \glscombinedsep is used.

These commands may be redefined as required. For example, to have no space between two elements that have both been marked as used and are both abbreviations (disregarding category attributes):
7. Multi (or Compound) Entries

There are some commands for redefining the above separators to common combinations.

\glssetcombinedsepabbrvnbsp

This does the following:

\glssetcombinedsepabbrvnone

This uses a non-breaking space (−) following an abbreviation (that has already been marked as used). Note that if the associated attributes are set the commands will behave according to the attribute.
This does nothing if either element are abbreviations that have already been used. Note that if the associated attributes are set the commands will behave according to the attribute.

\texttt{\renewcommand*\{\glscombinedsep\}[2]{{%\glshasattribute\{#1\}{combinedsep}\
\glshasattribute\{#1\}{combinedsep}}\%
{\ifglshasshort\{#1\}\{\ifglshasshort\{#2\}\{  \}}\}}%}

\texttt{\renewcommand*\{\glscombinedsepfirst\}[2]{{%\glshasattribute\{#1\}{combinedsepfirst}\
\glshasattribute\{#1\}{combinedsepfirst}}\%
{\ifglshasshort\{#1\}\{  \}}%}

\texttt{\renewcommand*\{\glscombinedfirstsep\}[2]{{%\glshasattribute\{#1\}{combinedfirstsep}\
{\ifglshasshort\{#2\}\{  \}}%}
}

\texttt{\renewcommand*\{\glscombinedfirstsepfirst\}[2]{{%\glshasattribute\{#1\}{combinedfirstsepfirst}\
{\ifglshasshort\{#1\}\{  \}}%}
}\
\glssetcombinedsepnarrow{\langle width\rangle}{\langle narrow-sep\rangle}

This is rather more complicated as it measures a field value and uses \langle narrow-sep\rangle if the width is less than \langle width\rangle. The field value is determined as follows:

- on first use the \texttt{long} field is used if it is set otherwise the \texttt{first} field is used;
- otherwise the \texttt{short} field is used if it is set otherwise the \texttt{text} field is used;

Note that this doesn’t take into account fonts, hooks, abbreviation styles or plural forms (e.g. \texttt{\mglspl}) or other field references (e.g. \texttt{\mglsname}). If the associated attributes are set the commands will behave according to the attribute.
7.5. \texttt{\textbackslash mgl\textbackslash l\textbackslash s} Element Hooks

The \texttt{\textbackslash mgl\textbackslash l\textbackslash s}-like commands use the following hooks:

\begin{itemize}
  \item \texttt{\textbackslash mgl\textbackslash s\textbackslash element\textbackslash prehook}
    This is done before each (non-skipped) element. (Default does nothing.)
  \item \texttt{\textbackslash mgl\textbackslash s\textbackslash element\textbackslash posthook}
    This is done after each (non-skipped) element. (Default does nothing.) Note that this is different from the normal entry post-link hook \texttt{\textbackslash gl\textbackslash s\textbackslash post\textbackslash link\textbackslash hook}. If the individual entry post-link hook is enabled (see the \texttt{post\textbackslash links} key in §7.9), this will go before \texttt{\textbackslash mgl\textbackslash s\textbackslash element\textbackslash posthook}.

    The definitions of the following commands are scoped within the \texttt{\textbackslash mgl\textbackslash s}-like commands so they can’t be accessed elsewhere (including in the post-link hook, see §7.6). They may be used in the above hooks or in the separator commands (described in §7.4) or in the command used to encapsulate the entire content. They can also be used in the post-link hook (see §5.5) to determine if an entry is being used within a \texttt{\textbackslash mgl\textbackslash s}-like command.
  \item \texttt{\textbackslash mgl\textbackslash s\textbackslash current\textbackslash multilabel}
    Expands to the multi-entry label.
  \item \texttt{\textbackslash mgl\textbackslash s\textbackslash current\textbackslash mainlabel}
    Expands to the label of the main element.
  \item \texttt{\textbackslash mgl\textbackslash s\textbackslash current\textbackslash list}
    Expands to the complete comma-separated list of elements.
  \item \texttt{\textbackslash mgl\textbackslash s\textbackslash current\textbackslash options}
    Expands to the options used when the multi-entry was defined. This doesn’t include options set with \texttt{\textbackslash multiglossary\textbackslash entry\textbackslash setup} or those passed to \texttt{\textbackslash mgl\textbackslash s} (or whichever variant is
being used).

\mglscurrentcategory
Expands to the multi-entry category current in effect.

\glsxtrcurrentmglscsname
Expands to the control sequence name of the calling command (for example, mgls or mglsp1).
To test if the current multi-entry is the first use:

\mglsisfirstuse{⟨true⟩}{⟨false⟩}
This does ⟨true⟩ if this is the first use otherwise it does ⟨false⟩. Note that this applies to the multi-entry first use flag not the first use flags of the individual elements.
At each iteration of the loop over the element list, the following commands are set, which can be accessed in hooks such as \mglselementprehook or in hooks used by the underlying \gls etc commands. For example, if \mglscurrentlabel is defined then \gls is being used inside \mgls.

\mglscurrentlabel
Expands to the current element label.

\mglselementindex
This is a count register that is set to the element index.

\mglscurrentprefix
Expands to the current multi-entry prefix.

\mglscurrentsuffix
7. Multi (or Compound) Entries

Expands to the current multi-entry suffix.

\mglslast\langle true\rangle\{false\}

If this is the last iteration, does \langle true\rangle otherwise does \langle false\rangle. This takes the skip options into account, so the last iteration may not necessarily be when the element index is equal to the total number of elements.

7.6. Post-Link Hook

There is a hook that occurs at the end the \mglslike commands according to the mpostlink setting (see §7.9). The hook used depends on the mpostlinkelement option. These hooks can’t access the commands described in §7.5 as the hook occurs outside of the scope in which they are defined.

The mpostlinkelement=custom option uses:

\mglscustompostlinkhook

This does nothing by default.

The mpostlinkelement=last option uses:

\mglslastelementpostlinkhook

which emulates the post-link hook of the last element.

The mpostlinkelement=main option uses:

\mglslastmainpostlinkhook

which emulates the post-link hook of the main element.

The default settings postlinks=none, mpostlink=true, mpostlinkelement=last will suppress the individual element post-link hooks (\glspostlinkhook) and do the multi-entry post-link hook for the last element (\mglslastelementpostlinkhook).

If you have the final element’s post-link hook enabled and the multi-entry post-link hook enabled (for example, postlinks=all, mpostlink=true, mpostlinkelement=last), the final element’s post-link hook will be done twice. Similarly for the main element with postlinks=all, mpostlink=true, mpostlinkelement=main.

The following commands are available for use in these hooks and may also be used in the

352
definition of \glsuffix.

\glslastmultilabel

Expands to the multi-entry label.

\glslastcategory

Expands to the multi-entry category (see §7.8). This will be empty if no category was assigned.

\glswasfirstuse{⟨true⟩}{⟨false⟩}

If that was the first use of the multi-entry (see §7.7) this does ⟨true⟩ otherwise it does ⟨false⟩.

### 7.6.1. Last Element

The following commands relate to the last element.

\glslastelementlabel

Expands to the label of the last non-skipped element. If all elements were skipped or if the multi-entry wasn’t defined, this will be empty.

Test if the last element was skipped:

\glsiflastelementskipped{⟨true⟩}{⟨false⟩}

If the last element was skipped this does ⟨true⟩ otherwise it does ⟨false⟩. If all elements were skipped or if the multi-entry wasn’t defined, this will do ⟨true⟩.

Test if the last element was its first use:

\glsiflastelementwasfirstuse{⟨true⟩}{⟨false⟩}

If the last non-skipped element was used for the first time this does ⟨true⟩ otherwise it does ⟨false⟩. (Corresponds to \glsxtrifwasfirstuse.) If all elements were skipped or if the multi-entry wasn’t defined, this will do ⟨true⟩.
7. Multi (or Compound) Entries

Test if the last element was plural:

\mglslastelementwasplural\{true\}\{false\}

If the last non-skipped element had the plural form displayed, this does \textit{true} otherwise it does \textit{false}. (Corresponds to \texttt{\glsifplural}.) If all elements were skipped or if the multi-entry wasn’t defined, this will do \textit{false}.

Test if the last element was had any case-changing applied:

\mglslastelementcapscase\{\textit{no-change}\}\{\textit{firstuc}\}\{\textit{all caps}\}

Corresponds to \texttt{\gls{capscase}} of the last non-skipped element. If all elements were skipped or if the multi-entry wasn’t defined, this will do \textit{no-change}.

7.6.2. Main Element

The following commands relate to the main element.

\mglslastmainlabel

Expands to the label of the main element from the multi-entry that was just referenced. If the main element was skipped or if the multi-entry wasn’t defined, this will be empty. If this is the same as \texttt{\mglslastelementlabel} then the main element was the last element.

Test if the main element was skipped:

\mglslastmainskipped\{true\}\{false\}

If the main element from the multi-entry that was just referenced was skipped this does \textit{true} otherwise it does \textit{false}. If the multi-entry wasn’t defined, this will do \textit{true}.

Test if the main element was its first use:

\mglslastmainwasfirstuse\{true\}\{false\}

If the main element was used for the first time this does \textit{true} otherwise it does \textit{false}. (Corresponds to \texttt{\glsxtrif{wasfirstuse}}.) If the main element was skipped or if the multi-entry wasn’t defined, this will do \textit{true}.

354
7. Multi (or Compound) Entries

Test if the main element was plural:

\mglstestplural{\text{true}}{\text{false}}

If the main element from the multi-entry that was just referenced had its plural form displayed this does \text{true} otherwise it does \text{false}. (Corresponds to \text{glsifplural}.) If the main element was skipped or if the multi-entry wasn’t defined, this will do \text{false}.

Test if the main element was had any case-changing applied:

\mglstestcapscase{\text{no-change}}{\text{firstuc}}{\text{all caps}}

Corresponds to \text{glscapscase} of the main element from the multi-entry that was just referenced. If the main element was skipped or if the multi-entry wasn’t defined, this will do \text{no-change}.

### 7.7. Multi-Entry First Use

Each multi-entry set has an associated first use flag. This is independent of the first use flag associated with the individual entries that make up the set. As with the \text{gls}-like commands, \text{mglstest} unsets this flag. Unlike the \text{glstext}-like commands, all the commands described in §7.11 (including commands like \text{mglstestname}) unset this flag, even if the elements use commands like \text{glsname} that don’t unset the entry’s first use flag.

You can determine whether or not a multi-entry set has been marked as used with:

\ifmglstest{\text{multi-label}}{\text{true}}{\text{false}}

This does \text{true} if the given multi-entry has been marked as used, otherwise it does \text{false}.

You can (globally) unset the flag (mark the set as having been referenced) with:

\mglstestunset{\text{multi-label}}

or reset it with:

\mglstestreset{\text{multi-label}}
There are also local versions of these commands:

\mglslocalunset{⟨multi-label⟩}
which locally unsets the flag and

\mglslocalreset{⟨multi-label⟩}
which locally resets the flag.
It’s also possible to unset or reset all multi-entries.

\mglsunsetall
Unsets all multi-entries.

\mglsresetall
Resets all multi-entries.

Note that unsetting or resetting any of the individual element first use flags doesn’t affect the multi-entry flag. Similarly, unsetting or resetting the multi-entry flag doesn’t affect the first use flags of the individual elements.

7.8. Multi-Entry Category

A multi-entry set may have an associated category set using the \texttt{category} key described in §7.9. This isn’t set by default, but if it is set the category may have attributes set in the usual way. The multi-entry category is independent of the individual entry categories. The following attribute is recognised by commands like \texttt{\mgls}:

\texttt{\multiglossaryentry}{multioptions=⟨options⟩}

The value are the default options to apply to any multi-entry set with the given category. These can be overridden by the first optional argument of \texttt{\multiglossaryentry} or by the
7. Multi (or Compound) Entries

setup key in the first optional argument of commands like \gls.

\begin{itemize}
  \item \texttt{combinedfirstsep=⟨separator⟩}

  The separator to use for \texttt{gl\$combinedfirstsep}.
  \end{itemize}

\begin{itemize}
  \item \texttt{combinedfirstsepfirst=⟨separator⟩}

  The separator to use for \texttt{gl\$combinedfirstsepfirst}.
  \end{itemize}

\begin{itemize}
  \item \texttt{combinedsepfirst=⟨separator⟩}

  The separator to use for \texttt{gl\$combinedsepfirst}.
  \end{itemize}

\begin{itemize}
  \item \texttt{combinedsep=⟨separator⟩}

  The separator to use for \texttt{gl\$combinedsep}.

  Note that you can’t access the category or its attributes via the multi-entry label (for example, with \texttt{gl\$hasattribute}). If you need to access the current multi-entry’s category within any of the \texttt{g\$l\$s-like hooks} (§7.5), you can obtain the category with \texttt{g\$l\$s\$current-category} and use commands like \texttt{gl\$shascategoryattribute}.
  \end{itemize}

7.9. Multi-Entry Settings

The settings that govern all multi-entries can be set using:

\begin{itemize}
  \item \texttt{\multiglossaryentrysetup{⟨options⟩}}

  The \langle options \rangle are the same as for \texttt{\multiglossaryentry}.

  Whenever the \texttt{g\$l\$s-like commands} are used, options are applied in the following order:

  1. general options identified by \texttt{\multiglossaryentrysetup};
  2. the category key is assigned if it’s in the general options, \texttt{\multiglossaryentry} or \texttt{setup} key;
  3. \texttt{multioptions} category attribute (if set);
  4. any options provided in the first optional argument of \texttt{\multiglossaryentry};
  \end{itemize}
7. Multi (or Compound) Entries

5. any options provided in the `setup` key in the first optional argument of the `\mgls`-like command.

These options are described below.

### 7.9.1. Indexing

- **indexmain**
  - \texttt{indexmain=⟨value⟩}
  - \texttt{default: true; initial: true}
  - This setting may take one of the following values:
    - `false` don’t index the main entry;
    - `true` index the main entry;
    - `first` only index the main entry if it’s the first use (of the main entry).

- **indexothers**
  - \texttt{indexothers=⟨value⟩}
  - \texttt{default: true; initial: true}
  - This setting may take one of the following values:
    - `false` don’t index the other entries;
    - `true` index the other entries;
    - `first` only index the other entries if it’s the first use (of the non-main entry).

### 7.9.2. Location Formats (Encaps)

- **encapmain**
  - \texttt{encapmain=⟨value⟩}
  - \texttt{initial: glsnumberformat}
  - This setting value should be the value to pass to `format` option (location encap) for the main entry.

- **encapothers**
  - \texttt{encapothers=⟨value⟩}
  - \texttt{initial: glsnumberformat}
  - This setting value should be the value to pass to `format` option (location encap) for the other (not main) entries.
7. Multi (or Compound) Entries

7.9.3. Post-Link Hooks

postlinks=⟨value⟩  \(\text{initial: none}\)

This setting determines whether or not to enable the individual element’s post-link hook. The value may be one of:

- none suppress the post-link hook for all elements;
- all don’t suppress the post-link hook for all elements;
- notlast only suppress the post-link hook for the last element;
- mainnotlast suppress the post-link hook for all “other” (not main) elements and for the last element (so only the main element will have its post-link hook as long as it’s not the last element);
- mainonly suppress the post-link hook for all “other” (not main) elements;
- othernotlast suppress the post-link hook for the main element and for the last element (so only the other elements will have their post-link hook as long as the element isn’t the last one);
- otheronly suppress the post-link hook for the main element.

mpostlink=⟨value⟩  \(\text{default: true}; \text{initial: true}\)

This setting determines whether or not to enable the multi-entry post-link hook (see §7.6). The value may be one of:

- false suppress the multi-entry post-link hook;
- true enable the multi-entry post-link hook;
- firstonly enable the multi-entry post-link hook only for the first use of the multi-entry;
- usedonly enable the multi-entry post-link hook only for the subsequent use of the multi-entry.

mpostlinkelement=⟨value⟩  \(\text{initial: last}\)

This setting indicates which post-link hook should be used if the multi-entry post-link hook has been enabled. Allowed values:
7. Multi (or Compound) Entries

last use \mglslastelementpostlinkhook (that is, use the post-link hook for the last element);

main use \mglslastmainpostlinkhook (that is, use the post-link hook for the main element);

custom use \mglscustompostlinkhook.

Some combinations may cause a repeated hook.

7.9.4. Prefixes and Suffixes

See §7.3 for more information on prefixes and suffixes. Note that the prefixes and suffixes are not affected by case-changing commands such as \MglS or \MGLS. If you want a prefix to obey case-changing, use the \mpglS-like commands instead (see §7.11.4).

\begin{itemize}
  \item \texttt{firstprefix} = \langle value \rangle
    \begin{itemize}
      \item The prefix to use on first use of the multi-entry.
    \end{itemize}
  \item \texttt{usedprefix} = \langle value \rangle
    \begin{itemize}
      \item The prefix to use on subsequent use of the multi-entry.
    \end{itemize}
  \item \texttt{firstsuffix} = \langle value \rangle
    \begin{itemize}
      \item The suffix to use on first use of the multi-entry.
    \end{itemize}
  \item \texttt{usedsuffix} = \langle value \rangle
    \begin{itemize}
      \item The suffix to use on subsequent use of the multi-entry.
    \end{itemize}
\end{itemize}

7.9.5. Skipping Elements

The skip options apply to the multi-entry first use flag not the individual element first use. See §7.7.
7. Multi (or Compound) Entries

\begin{align*}
\text{firstskipmain} &= \langle \text{boolean} \rangle \quad \text{default: true; initial: false} \\
\text{firstskipothers} &= \langle \text{boolean} \rangle \quad \text{default: true; initial: false} \\
\text{usedskipmain} &= \langle \text{boolean} \rangle \quad \text{default: true; initial: false} \\
\text{usedskipothers} &= \langle \text{boolean} \rangle \quad \text{default: true; initial: false}
\end{align*}

If true, the main element will be omitted on (multi-entry) first use.

If true, the other (non-main) elements will be omitted on (multi-entry) first use.

If true, the main element will be omitted on (multi-entry) subsequent use.

If true, the other (non-main) elements will be omitted on (multi-entry) subsequent use.

Note that it is technically possible to set the skip options so that both the main and the other elements are skipped. However, by default, this will generate a warning and only the final optional argument (the \langle insert\rangle) will be displayed. There won’t be a loop over all elements so the commands set at each iteration, such as \textbackslash mgl\textbackslash current\textbackslash label, won’t be defined.

The warning and the insertion of the \langle insert\rangle is done by:

\textbackslash g\textsl{xs}tr\textsl{gls}WarnAllSkipped\{\langle \text{message} \rangle\}\{\langle \text{insert} \rangle\}\{\langle \text{cs} \rangle\}

where \langle message\rangle is the warning message and \langle cs\rangle is the control sequence that encapsulates the entire content (including hyperlink and the \textsl{textformat} setting, if enabled).

If, for some particular reason, you want this scenario, you can redefine this command to omit the warning.

7.9.6. General

\begin{align*}
\text{hyper} &= \langle \text{value} \rangle \quad \text{initial: individual}
\end{align*}

This setting may take one of the following values:

- \texttt{none} no hyperlinks;
- \texttt{allmain} encapsulate the entire content with a single hyperlink to the main entry’s target;
- \texttt{mainonly} only hyperlink the main entry;
7. Multi (or Compound) Entries

individual hyperlink each entry individually;

otheronly only hyperlink the other entries;

notmainfirst don’t hyperlink the main entry on multi-entry first use;

nototherfirst don’t hyperlink the other entries on multi-entry first use;

notfirst don’t hyperlink any entries on multi-entry first use.

textformat=(value) initial: @firstofone

This setting value should be the control sequence name (without the leading backslash) of the command used to encapsulate the entire content.

category=(category-label)

The category to apply to the multi-entry. This is independent of the categories of each of the elements. The default is no category. See §7.8.

mglsopts=(option list)

Default options to pass to commands like \mglst. Note that setup can’t be used within this value.

7.10. \mglst Options

The ⟨options⟩ for \mglst (and similar commands) are listed below. Any additional options provided will be appended to the all value. For example:

\mglst[counter=chapter]{cbot}

is equivalent to:

\mglst[all=counter=chapter]{cbot}

Whereas:
7. Multi (or Compound) Entries

\mglscounter=chapter,all=counter=section\{cbot\}

is treated as:

\mglsaffect=counter=section,counter=chapter\{cbot\}

which has the same effect as:

\mglsaffect=counter=chapter\{cbot\}

The descriptions below reference \gls which is used internally by \mglsp. Replace this with \glspl as applicable for the variants, such as \glspl.

setup=⟨option list⟩

The value should be a list of any options that can be passed to \multiglossaryentrysetup. These options will override any conflicting options that were supplied to \multiglossaryentry or \multiglossaryentrysetup. Note that mglsopts can’t be used within this value.

all=⟨option list⟩

The value should be a list of any options that can be passed to \gls. These options will be passed to each instance of \gls and will override any conflicting setting in setup.

main=⟨option list⟩

The value should be a list of any options that can be passed to \gls. These options will be passed to the instance of \gls used for the main label and will override any conflicting setting in all.

others=⟨option list⟩

The value should be a list of any options that can be passed to \gls. These options will be passed to each instance of \gls used for the other (not main) labels and will override any
7. Multi (or Compound) Entries

conflicting setting in all.

hyper=\langle boolean \rangle  \quad default: true

A boolean option to determine whether or not to use hyperlinks (if supported). This may cause a conflict with other options, but is essentially provided to allow the starred version of \multicolumn to switch off all hyperlinks.

multiunset=\langle value \rangle  \quad initial: global

This only applies to the multi-entry first use flag, described in §7.7, not the first use flags of the elements. The value may be one of:

- global globally unset the flag;
- local locally unset the flag;
- none don’t unset the flag.

presetlocal=\langle boolean \rangle  \quad default: true; initial: false

A boolean option that governs whether or not the following options should have a local or global effect.

resetall=\langle boolean \rangle  \quad default: true; initial: false

A boolean option to determine whether or not to reset all elements before using \multicolumn. This option refers to the individual entry’s first use flag not the multi-entry first use flag. (This is similar to passing \prerest to each \multicolumn but it’s also applied to any skipped elements.)

resetmain=\langle boolean \rangle  \quad default: true; initial: false

A boolean option to determine whether or not to reset the main entry’s first use flag before using \multicolumn.

resetothers=\langle boolean \rangle  \quad default: true; initial: false
7. Multi (or Compound) Entries

A boolean option to determine whether or not to reset the first use flag of all the other (not main) elements before using \gls.

\begin{itemize}
  \item \texttt{unsetall}=(boolean) \quad default: \texttt{true}; initial: \texttt{false}
\end{itemize}

A boolean option to determine whether or not to unset all elements before using \gls. This option refers to the individual entry’s first use flag not the multi-entry first use flag. (This is similar to passing \texttt{preunset} to each \gls but it’s also applied to any skipped elements.)

\begin{itemize}
  \item \texttt{unsetmain}=(boolean) \quad default: \texttt{true}; initial: \texttt{false}
\end{itemize}

A boolean option to determine whether or not to unset the main entry’s first use flag before using \gls.

\begin{itemize}
  \item \texttt{unsetothers}=(boolean) \quad default: \texttt{true}; initial: \texttt{false}
\end{itemize}

A boolean option to determine whether or not to unset the first use flag of all the other (not main) elements before using \gls.

The reset... options all use:

\begin{itemize}
  \item \texttt{\mgls\elementreset}\{\textit{entry-label}\}
\end{itemize}

The unset... options all use:

\begin{itemize}
  \item \texttt{\mgls\elementunset}\{\textit{entry-label}\}
\end{itemize}

These take the \texttt{presetlocal} into account.

An alternative way of resetting the other elements is to use:

\begin{itemize}
  \item \texttt{\mgls\unsetothers}\{\textit{multi-label}\}
\end{itemize}

or for a local reset:

\begin{itemize}
  \item \texttt{\mgls\localunsetothers}\{\textit{multi-label}\}
\end{itemize}
7. Multi (or Compound) Entries

7.11. Variants of \texttt{mgls}

The commands listed in this section all behave like \texttt{mgls}. These (including \texttt{mgls} itself) are collectively referred to as the \texttt{mgls}-like commands. All commands unset the multi-entry first use flag (unless the \texttt{multiunset=none} option is applied). Only those commands that use the \texttt{gls}-like commands (such as \texttt{gls} or \texttt{glspl}) will unset the individual entry’s first use flag.

7.11.1. \texttt{gls}-like

\begin{itemize}
  \item \texttt{\mglsp{l}}\begin{itemize}[leftmargin=1.5em]
    \item \texttt{\langle options\rangle}\{\langle multi-label\rangle\}\{\langle insert\rangle\}
    \item \texttt{modifiers: \textit{* +} \langle alt-mod\rangle}
  \end{itemize}
  \begin{itemize}[leftmargin=1.5em]
  \item This uses \texttt{glsp1} instead of \texttt{gls} for each entry.
  \end{itemize}

  \item \texttt{\mglsmainpl}\begin{itemize}[leftmargin=1.5em]
    \item \texttt{\langle options\rangle}\{\langle multi-label\rangle\}\{\langle insert\rangle\}
    \item \texttt{modifiers: \textit{* +} \langle alt-mod\rangle}
  \end{itemize}
  \begin{itemize}[leftmargin=1.5em]
  \item This uses \texttt{glsp1} instead of \texttt{gls} for the main entry and \texttt{gls} for all the other entries.
  \end{itemize}

  \item \texttt{\Mgls}\begin{itemize}[leftmargin=1.5em]
    \item \texttt{\langle options\rangle}\{\langle multi-label\rangle\}\{\langle insert\rangle\}
    \item \texttt{modifiers: \textit{* +} \langle alt-mod\rangle}
  \end{itemize}
  \begin{itemize}[leftmargin=1.5em]
  \item This uses \texttt{Gls} for the first entry and \texttt{gls} for the other entries.
  \end{itemize}

  \item \texttt{\MGls}\begin{itemize}[leftmargin=1.5em]
    \item \texttt{\langle options\rangle}\{\langle multi-label\rangle\}\{\langle insert\rangle\}
    \item \texttt{modifiers: \textit{* +} \langle alt-mod\rangle}
  \end{itemize}
  \begin{itemize}[leftmargin=1.5em]
  \item This uses \texttt{Gls} for all entries.
  \end{itemize}

  \item \texttt{\Mglsp{l}}\begin{itemize}[leftmargin=1.5em]
    \item \texttt{\langle options\rangle}\{\langle multi-label\rangle\}\{\langle insert\rangle\}
    \item \texttt{modifiers: \textit{* +} \langle alt-mod\rangle}
  \end{itemize}
  \begin{itemize}[leftmargin=1.5em]
  \item This uses \texttt{glsp1} for the first entry and \texttt{glspl} for the remaining entries.
  \end{itemize}

  \item \texttt{\Mglsmainpl}\begin{itemize}[leftmargin=1.5em]
    \item \texttt{\langle options\rangle}\{\langle multi-label\rangle\}\{\langle insert\rangle\}
    \item \texttt{modifiers: \textit{* +} \langle alt-mod\rangle}
  \end{itemize}
  \begin{itemize}[leftmargin=1.5em]
  \item The first entry uses the appropriate sentence case command. The plural form is used for the main entry. So, if the first entry is also the main entry, \texttt{Glspl} is used, otherwise \texttt{Gls} is used. For the remaining entries, \texttt{glsp1} will be used if the entry is the main one, otherwise
  \end{itemize}
\end{itemize}
\gls will be used.

\MGlspl\{⟨options⟩\}{⟨multi-label⟩}\{⟨insert⟩\}     \text{modifiers: } * + \langle \text{alt-mod} \rangle
This uses \Glsp for all entries.

\MGlsmainpl\{⟨options⟩\}{⟨multi-label⟩}\{⟨insert⟩\}     \text{modifiers: } * + \langle \text{alt-mod} \rangle
This uses \Glsp for the main entry and \Gls for the other entries.

\MGLS\{⟨options⟩\}{⟨multi-label⟩}\{⟨insert⟩\}     \text{modifiers: } * + \langle \text{alt-mod} \rangle
This uses \GLS for all entries.

\MGLSp\{⟨options⟩\}{⟨multi-label⟩}\{⟨insert⟩\}     \text{modifiers: } * + \langle \text{alt-mod} \rangle
This uses \GLSp instead of \Gls for each entry.

\MGlsmainp\{⟨options⟩\}{⟨multi-label⟩}\{⟨insert⟩\}     \text{modifiers: } * + \langle \text{alt-mod} \rangle
This uses \GLSp for the main entry and \GLS for the others.

7.11.2. Abbreviations

\mglsshort\{⟨options⟩\}{⟨multi-label⟩}\{⟨insert⟩\}     \text{modifiers: } * + \langle \text{alt-mod} \rangle
This will use \glsxtrshort for any entries that have the \texttt{short} field set and will use \gls-
text otherwise.

\mglslong\{⟨options⟩\}{⟨multi-label⟩}\{⟨insert⟩\}     \text{modifiers: } * + \langle \text{alt-mod} \rangle
This will use \glsxtrlong for any entries that have the \texttt{long} field set and will use \glstext otherwise.

\mglssfull\{⟨options⟩\}{⟨multi-label⟩}\{⟨insert⟩\}     \text{modifiers: } * + \langle \text{alt-mod} \rangle
7. Multi (or Compound) Entries

This will use \glsxtrfull for any entries that have the short field set and will use \glsfirst otherwise.

\Mglsshort[(options)]{⟨multi-label⟩}[⟨insert⟩]  modifiers: * +  ⟨alt-mod⟩

As \mglsshort but sentence case for the first entry.

\Mglslong[(options)]{⟨multi-label⟩}[⟨insert⟩]  modifiers: * +  ⟨alt-mod⟩

As \mglslong but sentence case for the first entry.

\Mglssfull[(options)]{⟨multi-label⟩}[⟨insert⟩]  modifiers: * +  ⟨alt-mod⟩

As \mglssfull but sentence case for the first entry.

7.11.3. Other Fields

\glsname[(options)]{⟨multi-label⟩}[⟨insert⟩]  modifiers: * +  ⟨alt-mod⟩

This uses \glsname for each entry.

\Glsname[(options)]{⟨multi-label⟩}[⟨insert⟩]  modifiers: * +  ⟨alt-mod⟩

This uses \Glsname for the first entry and \glsname for the remaining entries.

\MGlsname[(options)]{⟨multi-label⟩}[⟨insert⟩]  modifiers: * +  ⟨alt-mod⟩

This uses \MGlsname for each entry.

\glssymbol[(options)]{⟨multi-label⟩}[⟨insert⟩]  modifiers: * +  ⟨alt-mod⟩

This uses \glssymbol for each entry if the symbol field has been set, otherwise it uses \gls.

\MGlssymbol[(options)]{⟨multi-label⟩}[⟨insert⟩]  modifiers: * +  ⟨alt-mod⟩
7. Multi (or Compound) Entries

This uses \glssymbol if the symbol field has been set otherwise it uses \Gls for each element. (Note that no case change is applied to the symbol as this usually isn’t appropriate.)

\Mglssymbol[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩

As \MGlssymbol, but \Gls is only used for the first element (if it doesn’t have the symbol field set).

\mglssusefield[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩

If the field given by \mgsfield exists for an element, \glsdisp will be used for that element, with the link text obtained from the field value (followed by the ⟨insert⟩), otherwise \gls will be used.

\mglssfield initial: useri

This command expands to the internal field label required by \mglssusefield. The default value is useri, which corresponds to the user1 key.

\Mglssusefield[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩

As \mglssusefield but sentence case for the first element.

\MGlssusefield[⟨options⟩]{⟨multi-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩

As \mglssusefield but sentence case for each element.

You can use the pre-element hook \mglselementprehook to locally redefine \mgsfield.

Examples:

1. if the multi-category is “sample” then use the user2 field otherwise use the user1 field:

\renewcommand{\mglselementprehook}{%  \ifdefstring{\mglscurrentcategory}{sample}{%    {\renewcommand{\mgsfield}{userii}}%  {\renewcommand{\mgsfield}{useri}}%}  \renewcommand{\mgsfield}{useri}}%}
2. if the element’s category is “sample” then use the \texttt{user2} field otherwise use the \texttt{user1} field:

\begin{verbatim}
\renewcommand{\mglselementprehook}{%
  \glsifcategory{\mglscurrentlabel}{sample}%
  {\renewcommand{\mglfield}{userii}}%
  {\renewcommand{\mglfield}{useri}}%
}
\end{verbatim}

3. if either the multi-entry’s category or the element’s category has the custom attribute “\texttt{mglfield}” set then use it otherwise use the \texttt{user1} field:

\begin{verbatim}
\renewcommand{\mglselementprehook}{%
  \glshascategoryattribute{\mglscurrentcategory}{mglfield}%
  {\renewcommand{\mglfield}{\glsgetcategoryattribute{\mglscurrentcategory}{mglfield}}}%
  {%
    \glshasattribute{\mglscurrentlabel}{mglfield}%
    {\renewcommand{\mglfield}{\glsgetattribute{\mglscurrentlabel}{mglfield}}}%
    {\renewcommand{\mglfield}{useri}}%
  }%
}
\end{verbatim}

7.11.4. Support for glossaries-prefix (\texttt{\pgls})

If you load the glossaries-prefix package (either after glossaries-extra) or with the \texttt{prefix} package option, then the following commands will use one of the \texttt{\pgls}\-like commands provided by that package. (See the glossaries user manual for further details.) If the glossaries-prefix package hasn’t been loaded then \texttt{\gls} (or analogous case-changing variant) will be used instead and a warning is issued with:

\begin{verbatim}
\mpglsWarning
\end{verbatim}

This may be redefined to do nothing to remove the warning.

\begin{verbatim}
\mpgls[(options)]{(multi-label)}[(insert)] modifiers: * + (alt-mod)
\end{verbatim}
7. Multi (or Compound) Entries

Uses \texttt{\textbackslash pgls} for the first element and \texttt{\textbackslash gls} for the remaining elements.

\begin{verbatim}
\texttt{\textbackslash mpglsp1[\langle options \rangle]\{\langle multi-label \rangle\}\{\langle insert \rangle\}}
\texttt{ modifiers: * + \langle alt-mod \rangle}
\end{verbatim}

Uses \texttt{\textbackslash pglspl} for the first element and \texttt{\textbackslash g1sp1} for the remaining elements.

\begin{verbatim}
\texttt{\textbackslash mpglsmainpl[\langle options \rangle]\{\langle multi-label \rangle\}\{\langle insert \rangle\}}
\texttt{ modifiers: * + \langle alt-mod \rangle}
\end{verbatim}

Only uses the plural form for the main element. The first element uses the prefixing command \texttt{\textbackslash pgls} or \texttt{\textbackslash pglspl}, depending on whether the first element is the main element.

\begin{verbatim}
\texttt{\textbackslash Mpgls[\langle options \rangle]\{\langle multi-label \rangle\}\{\langle insert \rangle\}}
\texttt{ modifiers: * + \langle alt-mod \rangle}
\end{verbatim}

Uses \texttt{\textbackslash Pgls} for the first element and \texttt{\textbackslash Gls} for the remaining elements.

\begin{verbatim}
\texttt{\textbackslash Mpglsmainpl[\langle options \rangle]\{\langle multi-label \rangle\}\{\langle insert \rangle\}}
\texttt{ modifiers: * + \langle alt-mod \rangle}
\end{verbatim}

Only uses the plural form for the main element. The first element uses the prefixing command \texttt{\textbackslash Pgls} or \texttt{\textbackslash Pglspl}, depending on whether the first element is the main element.

\begin{verbatim}
\texttt{\textbackslash MPGls[\langle options \rangle]\{\langle multi-label \rangle\}\{\langle insert \rangle\}}
\texttt{ modifiers: * + \langle alt-mod \rangle}
\end{verbatim}

Uses \texttt{\textbackslash Pgls} for the first element and \texttt{\textbackslash Gl1s} for the remaining elements.

\begin{verbatim}
\texttt{\textbackslash MPGlspl[\langle options \rangle]\{\langle multi-label \rangle\}\{\langle insert \rangle\}}
\texttt{ modifiers: * + \langle alt-mod \rangle}
\end{verbatim}

Uses \texttt{\textbackslash Pg1sp1} for the first element and \texttt{\textbackslash Gl1sp1} for the remaining elements.

\begin{verbatim}
\texttt{\textbackslash MPGlsmainpl[\langle options \rangle]\{\langle multi-label \rangle\}\{\langle insert \rangle\}}
\texttt{ modifiers: * + \langle alt-mod \rangle}
\end{verbatim}

Only uses the plural form for the main element. The first element uses the prefixing command \texttt{\textbackslash Pgls} or \texttt{\textbackslash Pglspl}, depending on whether the first element is the main element). All
7. Multi (or Compound) Entries

elements are use sentence case.

\MPGLS\[\langle options\rangle\]{\langle multi-label\rangle}\[\langle insert\rangle\] \textit{modifiers:} \* + \langle alt-mod\rangle

Uses \PGLS\ for the first element and \GLS\ for the remaining elements.

\MPGLSpl\[\langle options\rangle\]{\langle multi-label\rangle}\[\langle insert\rangle\] \textit{modifiers:} \* + \langle alt-mod\rangle

Uses \PGLSpl\ for the first element and \GLSpl\ for the remaining elements.

\MPGLSmainpl\[\langle options\rangle\]{\langle multi-label\rangle}\[\langle insert\rangle\] \textit{modifiers:} \* + \langle alt-mod\rangle

Only uses the plural form for the main element. All elements are converted to all caps. The first element uses the prefixing command (\PGLS\ or \PGLSpl\, depending on whether the first element is the main element).

7.12. Cross-References

Multi-entry labels may be used in the cross-referencing keys \textit{see} and \textit{seealso}. The formatting command will use:

\mglssseefirstitem\{\langle multi-label\rangle\}

for the first item in the list (if it's a multi-entry) and

\mglssseeitem\{\langle multi-label\rangle\}

for any subsequent items that are multi-entries. The default definition of \mglssseeitem\ is:

\newcommand*{\mglssseeitem}{\mglisname[all=noindex, setup={hyper=allmain}]{\#1} %
} %

This switches off indexing, sets the hyperlink to encompass the entire multi-entry content and uses the \textit{name} field. The default definition of \mglssseefirstitem\ is simply \mglssseeitem. For example, to use the \textit{short} or \textit{text} fields:
7. Multi (or Compound) Entries

A multi-entry label may also be used in the alias key. The hyperlink target will be the target for the main entry. For example:

\multiglossaryentry{cbot}{clostridium,botulinum}
\newglossaryentry{botox}{name={botox},description={},alias={cbot}}

In this case \gls{botox} will hyperlink to the botulinum target.

Any multi-entries used in the cross-referencing keys must be defined before the glossary is displayed. There is some support for docdef=true but not for the other docdef settings.

7.13. Additional Commands

You can test if a label represents a multi-entry using:

\glsxtrifmulti{⟨multi-label⟩}{⟨true⟩}{⟨false⟩}

This does ⟨true⟩ if a multi-entry has been defined with the label ⟨multi-label⟩ otherwise it does ⟨false⟩.

\glsxtrmultimain{⟨multi-label⟩}

Expands to the main entry label for the multi-entry identified by ⟨multi-label⟩ or nothing if not defined.

\glsxtrmultilist{⟨multi-label⟩}

Expands to the list of element labels for the multi-entry identified by ⟨multi-label⟩ or nothing if not defined.

\mglssforelements{⟨multi-label⟩}{⟨cs⟩}{⟨body⟩}
Iterates over all the list of element labels for the multi-entry identified by \textit{⟨multi-label⟩}. This defines ⟨cs⟩ to the current element label on each iteration of the loop, which can be used to reference the label in ⟨body⟩. This internally uses \@for (patched by the xfor package, which allows the loop to be broken).

\begin{verbatim}
\mglsfrothereelements{⟨multi-label⟩}{⟨cs⟩}{⟨body⟩}
\end{verbatim}

As \mglsforelements but skips the main entry label.

\begin{verbatim}
\glsxtrmultitotalelements{⟨multi-label⟩}
\end{verbatim}

Expands to the total number of elements in the given multi-entry or nothing if ⟨multi-label⟩ hasn’t been defined.

\begin{verbatim}
\glsxtrmultimainindex{⟨multi-label⟩}
\end{verbatim}

Expands to the index of the main element in the given multi-entry or nothing if ⟨multi-label⟩ hasn’t been defined. Indexing starts from 1 for the first element.

\begin{verbatim}
\glsxtrmultilastotherindex{⟨multi-label⟩}
\end{verbatim}

Expands to the index of the final non-main element in the given multi-entry or nothing if ⟨multi-label⟩ hasn’t been defined.

The \multiglossaryentry command will write the label information to the aux file using:

\begin{verbatim}
\writemultiglossentry{⟨options⟩}{⟨multi-label⟩}{⟨main-label⟩}{⟨list⟩}
\end{verbatim}

This is will write the following line to the aux file:

\begin{verbatim}
\@glsxtr@multientry{⟨options⟩}{⟨multi-label⟩}{⟨main-label⟩}{⟨list⟩}
\end{verbatim}

This is provided to support \texttt{docdef=true} and also for the benefit of any tools that require the information (such as \texttt{bib2gls} or autocompletion tools). If it’s not required and causes too much clutter, it can be disabled by redefining \writemultiglossentry to do nothing.
7. Multi (or Compound) Entries

7.14. bib2gls

In the bib2gls v2.9+ user manual, these multi-entry sets are referred to as “compound entries” or “compound sets” to differentiate them from bib2gls’s multi-entry types (such as @dualentry).

Each instance of one of the \mglslike commands is written to the aux file and so can be picked up by bib2gls (at least version 2.9). The resource option can be used to determine whether or not to consider the other (non-main) elements to be dependent on the main element.

With bib2gls, you can either define the compound entries in the document with \multiglossaryentry (or \providemultiglossaryentry) or you can use the @compoundset entry type in a bib file. Whichever method you use, remember that the entries that form the elements of the set must be defined first. See the bib2gls manual (v2.9+) for further details.

You can use the resource option compound-adjust-name to replace the name field of the main entry to:

\glsxtrmultientryadjustedname{\langle sublist1 \rangle}{\langle name \rangle}{\langle sublist2 \rangle}{\langle multi-label \rangle}

where ⟨multi-label⟩ is the label identifying the compound set, ⟨name⟩ was the value of the name before adjustment, ⟨sublist1⟩ is the sub-list of other element labels before the main element (empty if the main element is the first element in the list), and ⟨sublist2⟩ is the sub-list of other elements after the main element (empty if the main label is the final element).

This command is defined in glossaries-extra-bib2gls, which is automatically loaded with \record=only and \record=nameref. Case-changing versions of this command are also available.

\Glsxtrmultientryadjustedname{\langle sublist1 \rangle}{\langle name \rangle}{\langle sublist2 \rangle}{\langle multi-label \rangle}

This is a sentence case version of \glsxtrmultientryadjustedname.

\GlsXtrmultientryadjustedname{\langle sublist1 \rangle}{\langle name \rangle}{\langle sublist2 \rangle}{\langle multi-label \rangle}

This is a title case version of \glsxtrmultientryadjustedname.

\GLSxtrmultientryadjustedname{\langle sublist1 \rangle}{\langle name \rangle}{\langle sublist2 \rangle}{\langle multi-label \rangle}
7. Multi (or Compound) Entries

This is an all caps version of \texttt{\glsxtrmultientryadjustedname}. Note that the above commands don’t take the prefix or suffix into account (see §7.3). The separator between each element in the sub-lists is produced with:

\begin{verbatim}
\glsxtrmultientryadjustednamesep\{(pre-label)\}\{\(post-label\)\}
\end{verbatim}

The default definition just uses \texttt{\glscombinedfirstsepfirst}. The separator between the last element of \texttt{\(sublist1\)} and the main element is produced with:

\begin{verbatim}
\glsxtrmultientryadjustednamepresep\{(pre-label)\}\{\(post-label\)\}
\end{verbatim}

Similarly for the separator between the main element and the first element of \texttt{\(sublist2\)}:

\begin{verbatim}
\glsxtrmultientryadjustednamepostsep\{(pre-label)\}\{\(post-label\)\}
\end{verbatim}

These both default to \texttt{\glsxtrmultientryadjustednamesep}. The \texttt{\(name\)} is encapsulated with:

\begin{verbatim}
\glsxtrmultientryadjustednamefmt\{\(text\)\}
\end{verbatim}

This just does its argument by default. If \texttt{\(sublist1\)} is empty for the sentence case version, then \texttt{\(name\)} is encapsulated with:

\begin{verbatim}
\Glxsxtrmultientryadjustednamefmt\{\(text\)\}
\end{verbatim}

This does \texttt{\makefirstuc\{text\}} by default. For the title case version, the name is encapsulated with:

\begin{verbatim}
\Glsxtrmultientryadjustednamefmt\{\(text\)\}
\end{verbatim}

This uses \texttt{\glscapitalisewords}, if defined, or \texttt{\capitalisewords} otherwise. The all caps version uses:

\begin{verbatim}
\GLSxtrmultientryadjustednamefmt\{\(text\)\}
\end{verbatim}

This uses \texttt{\mfirstucMakeUppercase} by default.
Each element label in the sub-lists is encapsulated with:

\glsxtrmultientryadjustednameother{⟨text⟩}

This does \glsentryname{⟨label⟩} by default. For the sentence case version (where ⟨sub-list⟩ isn’t empty), then the element label is encapsulated with:

\Glsxtrmultientryadjustednameother{⟨text⟩}

This does \Glsentryname{⟨label⟩} by default. The title case version uses:

\GlsXtrmultientryadjustednameother{⟨text⟩}

This does \glsentrytitlecase{⟨label⟩}{name} by default. The all caps version uses:

\GLSxtrmultientryadjustednameother{⟨text⟩}

This is defined as:

\newcommand*{\GLSxtrmultientryadjustednameother}[1]{\% \mfirstucMakeUppercase{\glsentryname[#1]}}
8. Defining and Displaying Glossaries

As with the base glossaries package, you need to establish the indexing method in the preamble and use the appropriate \print⟨⟨...⟩⟩glossary command. For example, to use the “noidx” method you need \makenoidxglossaries in the preamble and \printnoidxglossary in the document. Whereas if you want to use makeindex or xindy, you need \makeglossaries in the preamble and \printglossary in the document. The glossaries-extra package provides a hybrid approach:

\makeglossaries[⟨⟨types⟩⟩]

If the optional argument is present, then ⟨⟨types⟩⟩ should be a comma-separated list of glossary labels that should be processed by makeindex or xindy as per the normal behaviour of \makeglossaries. Any non-ignored glossaries that are not listed in ⟨⟨types⟩⟩ should be treated as though \makenoidxglossaries was used. This means that the glossaries listed in ⟨⟨types⟩⟩ should be displayed using \printglossary and the other (non-ignored glossaries) should be displayed with \printnoidxglossary. See sample-mixedsort.tex for an example.

If the optional argument is omitted, it will be treated as per the original \makeglossaries provided by the base glossaries package. If the optional argument is present, \glsindexing-setting will be set to makeindex-noidx or xindy-noidx, depending on whether makeindex or xindy should be used.

Glossaries can’t be defined after \makeglossaries. This ensures that all the required indexing files are opened. If you’re not using \makeglossaries, glossaries need to be defined before any entries that should belong to them are defined.

The base glossaries package provides \newignoredglossary to define an ignored glossary that doesn’t have any associated indexing files. This will automatically switch off hyperlinks for any entries assigned to the glossary (since there will be no target). With glossaries-extra, it’s possible to have targets without using the indexing methods provided by the base package. For example, it’s possible to have standalone entries (see §8.5) or targets can be created with \printunsrtglossary, so glossaries-extra provides a starred version.

\newignoredglossary*{⟨⟨glossary-label⟩⟩}
This behaves like the unstarred version but doesn’t disable hyperlinks. The glossary will still be omitted by iterative commands, such as `\printunsrtglossaries`, and can’t be used with `\printglossary` or `\printnoidxglossary`. If you use an ignored glossary with `\printunsrtglossary`, you will need to use the `title` option to override the default title, if required.

\provideignoredglossary{⟨glossary-label⟩}

This is like `\newignoredglossary` but does nothing if the glossary has already been defined.

With the indexing options provided by the base glossaries package, if you want a term to appear in more than one glossary, it’s necessary to define a duplicate entry with a different label. With the “unsr” family of commands, the same entry can appear in multiple glossaries. This can be done by simply copying the entry’s label to the required glossary using:

\glsxtrcopytoglossary{⟨entry-label⟩}{⟨glossary-type⟩}

This just adds the label to the target glossary’s internal comma-separated list that commands like `\printunsrtglossary` iterate over. The unstarred version locally adds the label. The starred version performs a global change.

\glsxtrcopytoglossary is not compatible with `docdef=true`.

Note that the `type` field will still be set to the original glossary. This is considered the entry’s primary glossary. There’s no field that keeps track of the additional glossaries the entry has been copied to.

If used with `\printglossary` or `\printnoidxglossary`, the entry will only be indexed for its primary glossary. It won’t show up in the other glossaries, but will be found when using an iterative command, such as `\glsaddall`, over the target glossary.

You can test if an entry has already been added to a glossary with:

\GlsXtrIfInGlossary{⟨entry-label⟩}{⟨glossary-type⟩}{⟨true⟩}{⟨false⟩}

This does ⟨true⟩ if the entry given by ⟨entry-label⟩ is in the internal list of the glossary identified by ⟨glossary-type⟩, otherwise it does ⟨false⟩. If the glossary doesn’t exist, this does ⟨false⟩ and will either generate an error (`undefaction=error`) or a warning (`undefaction=warn`). This command considers ignored glossaries as existing.
You can test if a glossary is empty with:

\glsxtrifemptyglossary{⟨glossary-type⟩}{⟨true⟩}{⟨false⟩}

This does ⟨true⟩ if the glossary identified by ⟨glossary-type⟩ is empty, otherwise does ⟨false⟩. If the glossary doesn’t exist, this does ⟨true⟩ and will either generate an error (undefaction = error) or a warning (undefaction = warn). This command considers ignored glossaries as existing.

To test for the existence of a glossary, you can use \ifglossaryexists and \doifglossarynoexistsordo, which are documented in the “Conditionals” section of the glossaries user manual.

The base glossaries package provides \forallglossaries to iterate over a list of glossaries labels (all non-ignored glossaries by default). This can also be used with glossaries-extra but \forallacronyms is only for glossaries that have been declared as lists of acronyms, so it’s inappropriate with the glossaries-extra package. Instead, you can use the analogous command:

\forallabbreviationlists{⟨cs⟩}{⟨body⟩}

Each instance of \newabbreviation will add the abbreviation’s associated glossary (identified by the type key) to the internal list of labels (if not already added). Note that this won’t take into account any glossaries that had abbreviations copied or moved to it.

### 8.1. Entry Page Reference

The base glossaries package provides \glsrefentry, which uses \ref to reference the entry’s associated counter (enabled with entrycounter or subentrycounter, not the location counter). The glossaries-extra package additionally provides:

\glsxtrpageref{⟨entry-label⟩}

This works in the same way but uses \pageref instead of \ref. As with \glsrefentry, if the corresponding counter has not been enabled, this just does \gls{⟨entry-label⟩}.

### 8.2. Glossary Preamble

The base package provides \glossarypreamble, which is used at the start of the glossary. By default, this will use the preamble associated with the current glossary. If you redefine \glossarypreamble, this will set the preamble for all glossaries. To set the preamble for
8. Defining and Displaying Glossaries

a particular glossary, you can use \setglossarypreamble. With glossaries-extra, you can additionally append to an existing preamble using:

\apptoglossarypreamble\[⟨\text{type}⟩\]{⟨\text{text}⟩}

This (locally) appends ⟨text⟩ to the preamble for the glossary identified by ⟨type⟩. If ⟨type⟩ is omitted, \glsdefaulttype is assumed.

\pretoglossarypreamble\[⟨\text{type}⟩\]{⟨\text{text}⟩}

This (locally) prepends ⟨text⟩ to the preamble for the glossary identified by ⟨type⟩. If ⟨type⟩ is omitted, \glsdefaulttype is assumed.

8.3. Options

In addition to the options available with \printglossary, the following options are also provided. Some of these listed here are specific to \printunsrtglossary and printunsrtglossarywrap. Options provided by the base package that aren’t available for the “unsrt” family of commands are identified below.

sort=⟨\text{method}⟩

This option is only available for \printnoidxglossary. The \printunsrtglossary and \printunsrtinnerglossary commands simply iterate over the glossary’s internal list in the order in which the entries have been added to that glossary. If you are using bib2gls, use the sort resource option instead.

title=⟨\text{text}⟩

This option is provided by the base glossaries package to override the default title for the glossary. This option is also available for \printunsrtglossary and printunsrtglossarywrap but not for \printunsrtinnerglossary.

toctitle=⟨\text{text}⟩

This option is provided by the base glossaries package to override the default table of contents title for the glossary. This option is also available for \printunsrtglossary and printunsrt-
8. Defining and Displaying Glossaries

glossarywrap but not for \printunsrtinnerglossary.

\begin{itemize}
\item \texttt{numberedsection=⟨value⟩}
\textit{default: nolabel; initial: false}
\end{itemize}

This option is provided by the base glossaries package to indicate whether or not the section header used at the start of the glossary should be numbered rather than unnumbered (and whether or not to automatically label the glossary with \label{\glsautofromprefix\texttt{glossary-type}}). The \texttt{numberedsection} package option will change the default setting to match. This option is not available for \printunsrtinnerglossary.

\begin{itemize}
\item \texttt{style=⟨style-name⟩}
\end{itemize}

This option is provided by the base glossaries package to set the glossary style. This option is not available for \printunsrtinnerglossary.

\begin{itemize}
\item \texttt{label=⟨label⟩}
\end{itemize}

This option is provided by glossaries-extra to add \label{⟨label⟩} after the section header. This is essentially like \texttt{numberedsection=nameref} but you supply the label. This option is not available for \printunsrtinnerglossary. Alternatively, you can use:

\begin{itemize}
\item \texttt{\glsxtrsetglossarylabel{⟨label⟩}}
\end{itemize}

This will need to be scoped or changed between glossaries or use a command in ⟨label⟩ that expands differently for each glossary to avoid duplicate labels.

\begin{itemize}
\item If the supplied value is empty, the label is suppressed (without otherwise altering the \texttt{numberedsection} setting).
\end{itemize}

\begin{itemize}
\item \texttt{leveloffset=⟨offset⟩}
\textit{initial: 0}
\end{itemize}

This option sets or increments the hierarchical level offset. If ⟨offset⟩ starts with ++ then the current offset is incremented by the given amount otherwise the current offset is set to ⟨offset⟩. For example, an entry with a normal hierarchical level of 1 will be treated as though it has hierarchical level 1+⟨offset⟩. Note that the glossary style may not support the resulting hierarchical level. This option is only available for the “unsrt” family of commands and the
8. Defining and Displaying Glossaries

printunsrtglossarywrap environment. See §8.4.3.1 for an example.

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>flatten</td>
<td>boolean</td>
<td>true; false</td>
</tr>
</tbody>
</table>

Treats all entries as though they have the same hierarchical level (the value of leveloffset). This option is only available for the “unsrt” family of commands and the printunsrtglossarywrap environment. Unlike the flatten resource option, this option doesn’t actually remove the parent field.

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>boolean</td>
<td>true; true</td>
</tr>
</tbody>
</table>

This option is only applicable to the “unsrt” family of commands and printunsrtglossarywrap. If set to false, it will prevent groups from being formed. If true (the default), groups will only be formed if they are supported. See §8.4 for further details.

<table>
<thead>
<tr>
<th>Option</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td>⟨prefix⟩</td>
</tr>
</tbody>
</table>

This option is provided by glossaries-extra and simply redefines \glolinkprefix to expand to ⟨prefix⟩. If hyperlinks are supported and the glossary style uses \glstarget to create the entry’s hypertarget, the target name is obtained from \glolinkprefix⟨entry-label⟩. If you are displaying multiple glossaries with shared entries (for example, using the secondary resource option with bib2gls), then changing the prefix can avoid duplicate targets. Alternatively, you can redefine \glstarget to use \glsxtrtarget.

Note that this option will also affect the targets used by the \gls-like and \glstext-like commands. This means that if you have, for example, \gls in the description of an entry, then its hyperlink will go to that entry’s item in the current glossary. Whereas referencing that entry outside of the glossary will hyperlink to the glossary that uses the prefix matching the setting at that point in the document. For example:

```latex
\usepackage[colorlinks]{hyperref}
\usepackage[showtargets=annoteleft,style=tree]{glossaries-extra}
\newglossaryentry{sample}{name={sample},
  description={an example description that references \gls{another}}}
\newglossaryentry{another}{name={another},
  description={some other example description that references \gls{sample}}}
\begin{document}
Link to glossary 1: \gls{sample}.
```

383
8. Defining and Displaying Glossaries

Link to glossary 2: \gls[prefix=other-]{sample}.
\printunsrtglossary
\printunsrtglossary[prefix=other-]
\end{document}

This uses the \texttt{showtargets} package option to show the target names to the left of the hyperlink or hypertarget. The result is:

Example 132: Changing the target prefix

Link to glossary 1: [glo:sample]sample.  
Link to glossary 2: [other-sample]sample.

Glossary

[glo:sample]sample an example description that references [glo:another]another.

[glo:another]another some other example description that references [glo:sample]sample.

Glossary

[other-sample]sample an example description that references [other-another]another.

[other-another]another some other example description that references [other-sample]sample.

Within the main part of the document, the first reference to “sample” has a hyperlink to the first glossary (with the target glo:sample, which uses the default prefix), and the second reference has a hyperlink to the second glossary (with the target other-sample).

Within the glossaries, the \gls references use the current glossary prefix, so the target is in the same glossary.

targetnameprefix={prefix}

This is similar to the previous option but only affects the prefix for the entry item’s target and doesn’t change the prefix for any references contained within the glossary. This \textit{prepends} the given prefix to the default prefix.

If the above example is modified to:
8. Defining and Displaying Glossaries

Link to glossary 1: \gls{sample}.

Link to glossary 2: \gls[prefix=other-glo]{sample}.
\printunsrtglossary
\printunsrtglossary[targetnameprefix=other-]

Then the result will be:

Example 133: Prepending to the target prefix for just the entry item

Link to glossary 1: \[glo:sample\]sample\].
Link to glossary 2: \[other-glo:sample\]sample\].

Glossary

\[glo:sample\]sample\] an example description that references \[glo:another\]another\].
\[glo:another\]another\] some other example description that references \[glo:sample\]sample\].

Glossary

\[other-glo:sample\]sample\] an example description that references \[glo:another\]another\].
\[other-glo:another\]another\] some other example description that references \[glo:sample\]sample\].

Note that this has prepended other- to the existing glo: prefix. This is why the prefix option in the second \gls reference had to be changed to match the appropriate hypertarget name. The \gls references in the second glossary now point to the relevant line in the first glossary.

It’s possible to combine targetnameprefix with prefix={} but that will also affect the \gls references within the glossary.

\texttt{target=\langle boolean\rangle}
\hspace{1em} \textbf{default: true; initial: true}

This is a boolean option that can be used to switch off the automatic creation of the entry hypertargets but still allows hyperlinks within the glossary. This can be used to prevent
duplicate destinations for secondary glossaries.

\begin{verbatim}
preamble=(text)
\end{verbatim}

Redefines \texttt{\glossarypreamble} to \texttt{⟨text⟩}.

\begin{verbatim}
postamble=(text)
\end{verbatim}

Redefines \texttt{\glossarypostamble} to \texttt{⟨text⟩}.

## 8.4. Displaying a Glossary Without Sorting or Indexing

The base glossaries package provides two ways of displaying a glossary, depending on the indexing option: \texttt{\printglossary} (used with \texttt{\makeglossaries}) or \texttt{\printnoidxglossary} (used with \texttt{\makenoidxglossaries}).

The glossaries-extra package provides an alternative that doesn’t require sorting or indexing.

\begin{verbatim}
\printunsrtglossary[(⟨options⟩)]
\end{verbatim}

This behaves in a similar way to \texttt{\printnoidxglossary}, but it always lists all the defined entries for the given glossary in the order in which they were added to the glossary. Unlike \texttt{\printglossary}, you may use \texttt{\printunsrtglossary} with an ignored glossary.

The “unsrt” family of commands and \texttt{\printunsrtglossary} are not intended for use with \texttt{\makeglossaries} and \texttt{\makenoidxglossaries}. Mixing these different methods can result in unexpected behaviour.

There is also a starred version which has a mandatory argument:

\begin{verbatim}
\printunsrtglossary*[⟨⟨options⟩]⟨⟨init-code⟩⟩
\end{verbatim}

This is equivalent to:

\begin{verbatim}
\begingroup
  ⟨⟨init-code⟩⟩\printunsrtglossary[(⟨options⟩)]
\end{verbatim}

There’s no significant difference between doing:
8. Defining and Displaying Glossaries

\{⟨init-code⟩\printunsrtglossary[⟨options⟩]\}

and

\printunsrtglossary*[⟨options⟩]{⟨init-code⟩}

Note that unlike \glossarypreamble, the supplied \langleinit-code⟩ is done before the glossary header.

If you want to use one of the tabular-like styles with \printunsrtglossary, make sure you load glossaries-extra-stylemods which modifies the definition of \glsgroup-skip to avoid the “Incomplete \iftrue” error that may otherwise occur.

As with \printglossary and \printnoidxglossary, there is also a command to print all non-ignored glossaries in the order in which they were defined:

\printunsrtglossaries

This means you now have the option to simply list all entries on the first \LaTeX run without the need for a post-processor, however there will be no location list in this case, as that has to be set by a post-processor such as bib2gls (see §11).

No attempt is made to gather hierarchical elements. If child entries aren’t defined immediately after their parent entry, they won’t be together in the glossary when using \printunsrtglossary.

The way that \printunsrtglossary basically works is to iterate over every label in the glossary’s internal label list and format each entry according to the way the glossary style would normally format the entry’s hierarchical level (described in more detail in §8.4.3). If a change in letter group is detected, the letter group heading and group skip will be inserted.

A label is appended to the glossary’s internal label list whenever an entry is defined. This means that the list will normally be in order of definition, but it’s also possible to copy an entry’s label to another glossary’s internal label list using \glsxtrcopytoglossary, which can be used to provide a different order. A simple example:

\documentclass{article}
\usepackage[style=treegroup]{glossaries-extra}
\newglossaryentry{ant}{name={ant},description={}}
8. Defining and Displaying Glossaries

\newglossaryentry{gazelle}{name={gazelle},description={}}
\begin{document}
\printunsrtglossary
\end{document}

The document build only requires one \LaTeX call in this case.

Example 134: Displaying unsorted glossaries

Glossary

A
ant
G
gazelle

Note the difference if the stylemods option is used:

\usepackage[stylemods,style=treegroup]{glossaries-extra}

Example 135: Displaying unsorted glossaries with stylemods

Glossary

65
ant
71
gazelle

In this case, the group headings are now numbers instead of letters. The styles provided with glossaries-extra and those modified by glossaries-extra-stylemods are designed to assist integration with bib2gls. Without these modifications, \printunsrtglossary behaves like the less sophisticated \printnoidxglossary which checks if the label is an integer less than 256 and uses \char to create the title (if no title has been provided).

If you really want to use \printunsrtglossary without bib2gls and you want letter
groups with `stylemods` without having to define all the titles, you can use:

\glsxtrnoidxgroups

which will switch over to using the group titling method used with `\printnoidxglossary` (which only supports ASCII). This command is only available with `record=off` and can’t be used with `\makeglossaries`.

If, conversely, you don’t want any groups formed, regardless of the glossary style, you can disable them with `groups=false`. The above example can be modified so that the document contains:

\printunsrtglossary[title={Glossary 1}]
\glsxtrnoidxgroups
\printunsrtglossary[title={Glossary 2}]
\printunsrtglossary[groups=false,title={Glossary 3}]
\printunsrtglossary[style=tree,nogroupskip,title={Glossary 4}]

This repeats the same glossary. The first is the same as the previous example. The second is the same as the example that didn’t use `stylemods`. The final two glossaries have the groups suppressed. Using `groups=false` (Glossary 3) is more efficient than using `nogroupskip` and switching to a style that doesn’t show the header (Glossary 4).

I’ve also switched to two column mode to display the result in a more compact form. The first two glossaries are shown on the left and the last two are on the right:
8. Defining and Displaying Glossaries

Example 136: Displaying unsorted glossaries with different group settings

<table>
<thead>
<tr>
<th>Glossary 1</th>
<th>Glossary 2</th>
<th>Glossary 3</th>
<th>Glossary 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>A</td>
<td>ant</td>
<td>ant</td>
</tr>
<tr>
<td>ant</td>
<td>ant</td>
<td>gazelle</td>
<td>gazelle</td>
</tr>
<tr>
<td>71</td>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gazelle</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The “unsrt” family of commands were designed for use with \bib2gls, which uses more complex alphanumerical group labels to allow for greater customization and to avoid conflict where there are multiple glossaries or hierarchical levels with potentially the same letter groups.

The way that \bib2gls works is to select entries from a \bib file, according to the document requirements, sort the entries, and then write the entry definitions (with commands like \longnewglossaryentry* or \newabbreviation) in the \glstex in the desired order, which is then input by \GlsXtrLoadResources. This means that \printunsrtglossary will display the entries in that order since, from glossaries-extra’s point of view, that’s the order of definition.

While it is possible to use \printunsrtglossary without \bib2gls, as in the above example, for long or complex glossaries it’s better to use \bib2gls which can automatically assign appropriate titles to the groups.

Groups and hierarchy are discussed in more detail in §8.4.1. See §8.4.2 for location lists and §8.4.3.1. Advanced commands and further detail about the way \printunsrtglossary works are covered in §8.4.3.
8. Defining and Displaying Glossaries

8.4.1. Groups and Hierarchy

See Gallery: Logical Glossary Divisions (type vs group vs parent)\(^a\) for the difference between the group, type and parent fields.

\(^a\)dickimaw-books.com/gallery/index.php?label=logicaldivisions

Here’s a longer example that uses \texttt{stylemods} to automatically load glossary-bookindex:

\begin{verbatim}
\documentclass{article}
\usepackage[stylemods=bookindex,style=bookindex]{glossaries-extra}
\newglossaryentry{waterfowl}{name={waterfowl},description={}}
\newglossaryentry{ant}{name={ant},description={}}
\newglossaryentry{adder}{name={adder},description={}}
\newglossaryentry{duck}{name={duck},parent={waterfowl},description={}}
\newglossaryentry{zebra}{name={zebra},description={}}
\newglossaryentry{aardvark}{name={aardvark},description={}}
\newglossaryentry{gazelle}{name={gazelle},description={}}
\newglossaryentry{mallard}{name={mallard},parent={duck},description={}}
\newglossary*{another}{Another Glossary}
\glsxtrcopytoglossary{mallard}{another}
\glsxtrcopytoglossary{aardvark}{another}
\glsxtrcopytoglossary{zebra}{another}
\glsxtrcopytoglossary{ant}{another}
\glsxtrcopytoglossary{duck}{another}
\begin{document}
\printunsrtglossary
\printunsrtglossary[type=another]
\end{document}
\end{verbatim}

Unlike the previous examples that defined the entries in alphabetical order, this example hasn’t used any logical order. Note, in particular, that the child entries “duck” and “mallard” (which have the \texttt{parent} key set) have not been defined immediately after their parent.

The first \texttt{\printunsrtglossary} has the default \texttt{type=main} and lists all entries defined in the main glossary, in the order in which they were defined. The second \texttt{\printunsrtglossary} lists all entries in the custom another glossary and is in the order in which the entries were copied to that glossary.

The document build again simply requires one \LaTeX{} call. The result is shown in Example 137 on the following page.
8. Defining and Displaying Glossaries

Example 137: Displaying unsorted glossaries with a copied list

<table>
<thead>
<tr>
<th>Glossary</th>
</tr>
</thead>
<tbody>
<tr>
<td>waterfowl</td>
</tr>
<tr>
<td>ant</td>
</tr>
<tr>
<td>adder</td>
</tr>
<tr>
<td>duck</td>
</tr>
<tr>
<td>zebra</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Another Glossary</th>
</tr>
</thead>
<tbody>
<tr>
<td>mallard</td>
</tr>
<tr>
<td>aardvark</td>
</tr>
<tr>
<td>ant</td>
</tr>
<tr>
<td>duck</td>
</tr>
<tr>
<td>zebra</td>
</tr>
</tbody>
</table>

There are some oddities in both lists. It’s the glossary style that determines the formatting of the entries according to the entry’s hierarchical level, but it looks strange for the duck and mallard entries to be indented when they don’t follow after their parent entry.

As the internal loop within \printnordsrtglossary iterates over each entry, it tries to determine which letter group the entry belongs to. If it’s different from the group for the previous entry (in the same hierarchical level), a group header is added (which may or may not be displayed, depending on the glossary style). This means than an unordered list of entries, such as in the above example, may contain repeated headers.

The way that the group is determined depends on whether or not the group key has been defined. If it isn’t defined (the default), then the group label is obtained from the uppercase character code of the first token of the sort key. If the token doesn’t have an uppercase character code (indicating that it’s not a letter) or if the sort value is empty then the label will be set to glssymbols (which corresponds to the symbol group). This is the same way that \printnoidxglossary inserts groups.
Remember that if the `sort` key hasn’t been set, it will be assigned automatically to the same value as the `name` key (or with `sort=use` or `sort=def` to a numerical value). The `sort` key will be empty if you use `sort=clear`. The `sort=none` setting simply skips the pre-processing of the `sort` key (such as sanitizing).

For example, the `ant` entry doesn’t explicitly use the `sort` key, so the sort value is obtained from the `name` key, which is set to `ant`. The first token is “a”, which is a letter. The group’s label is obtained from the letter’s uppercase character decimal code (65). There’s no associated title (which can be assigned with \glsxtrsetgrouptitle), so the title is simply “65” (with `stylemods`, see earlier) or “A” (without `stylemods` or with \glsxtrnoidxgroups).

The `ant` entry is followed by “adder”. The same process determines that the “adder” group label is also 65. There’s no change in the group label from the previous entry (ant) so no header is inserted.

By default, this group check is omitted for child entries, which is why no group header is inserted before duck or mallard. So the next entry to be checked for a group is the zebra entry, which has the group label 90 (the decimal code for “Z”). Again there’s no title associated with that label so the title is simply the label.

The zebra entry is followed by aardvark which, following the same process, has the group label 65. This is different from the previous group label (90) so a group header is inserted. This is why there are two “90” letter groups.

---

The “unsrt” family of commands don’t order the entries.

If the `group` key has been defined (which is the case with `record=only` and `record=nameref`) then the group label is obtained from the `group` field. If the `group` field is defined but empty then the entry will belong to the empty group. The value of the `sort` field is now irrelevant.

So, simply adding the `record` option to the above example document will cause the group headers to disappear. This is because the `group` key will now be defined but is empty for each entry. Even with a style like `bookindex`, there won’t be any group headers.

Provided the `group` key has been defined, the field used to store the group label is given by:

\begin{verbatim}
\glsxtrgroupfield initial: group
\end{verbatim}

This expands to `group`, by default. However it’s possible to use a different field in which to store the group label, in which case \glsxtrgroupfield needs to be redefined. For example:

\begin{verbatim}
{\renewcommand{\glsxtrgroupfield}{othergroup}\printunsrtglossary}
\end{verbatim}

or

393
8. Defining and Displaying Glossaries

\printunsrtglossary*{\renewcommand{\glsxtrgroupfield}{othergroup}}

(but this still requires the group key to be defined, even if it’s not being used to store the group label). With bib2gls, the secondary resource option (combined with --group) will store the group label obtained from the secondary sort in the secondarygroup field and adds the redefinition of \glsxtrgroupfield to the associated glossary preamble. This prevents it from clashing with the group field in the event that the secondary sort method has produced a different set of groups (which is likely).

The follow example document uses record to create the group key and assigns group labels with associated titles. Note that the secondarygroup field doesn’t have an associated key, so it needs to be set with a field assignment command, such as $\GlsXtrSetField$.

\documentclass{article}
\usepackage[record,stylemods=bookindex,style=bookindex]{glossaries-extra}
\glsxtrsetgrouptitle{group1label}{Group 1}
\glsxtrsetgrouptitle{group2label}{Group 2}
\glsxtrsetgrouptitle{group3label}{Group 3}
\glsxtrsetgrouptitle{group4label}{Group 4}
\newglossaryentry{waterfowl}{name={waterfowl},description={},group={group1label}}
\newglossaryentry{ant}{name={ant},description={},group={group1label}}
\GlsXtrSetField{ant}{secondarygroup}{group4label}
\newglossaryentry{adder}{name={adder},description={},group={group2label}}
\GlsXtrSetField{adder}{secondarygroup}{group1label}
\newglossaryentry{duck}{name={duck},parent={waterfowl},description={},group={group4label}}
\GlsXtrSetField{duck}{secondarygroup}{group2label}
\newglossaryentry{zebra}{name={zebra},description={},group={group2label}}
\GlsXtrSetField{zebra}{secondarygroup}{group3label}
\newglossaryentry{aardvark}{name={aardvark},description={},group={group2label}}
\GlsXtrSetField{aardvark}{secondarygroup}{group1label}
\newglossaryentry{gazelle}{name={gazelle},description={},group={group1label}}
\GlsXtrSetField{aardvark}{secondarygroup}{group1label}
\newglossaryentry{gazelle}{name={gazelle},description={},group={group1label}}
This is essentially mimicking the way that the \texttt{secondary} resource option sets the \texttt{secondarygroup} field and adds the redefinition of \texttt{\glsxtrgroupfield} to the secondary glossary’s preamble. (Although in this case, there’s no logical order.) The result is shown in Example 138 on the next page.

Note that even though the duck and mallard entries have the \texttt{group} and \texttt{secondarygroup} fields set, there’s no group title for them in either glossary because they are child entries.

\begin{Verbatim}
\glsxtraddgroup{(entry-label)}{(code)}
\end{Verbatim}

This command will perform \texttt{(code)} if the entry identified by \texttt{(entry-label)} should have group support (provided the \texttt{group} field has been set). The default definition is:

\begin{Verbatim}
\newcommand*{\glsxtraddgroup}[2]{%
   \ifglshasparent{#1}{#2}%
   \else
      \ifglsxtrprintglossflatten
         \ifnum#2=2%
            #2%
         \else
            \fi
      \fi
   \fi
}
\end{Verbatim}

This means that only entries that don’t have a parent (with \texttt{flatten=false}) or any entry (with \texttt{flatten=true}) will have the group check performed. With \texttt{bib2gls}, the \texttt{group-level} will redefine \texttt{\glsxtraddgroup} to always do \texttt{(code)}, which means that all entries will have the group check performed.
Example 138: Displaying unsorted glossaries with custom groups

Glossary

Group 1
waterfowl
ant  

Group 2
adder
duck  

Another Glossary

mallard  

Group 1
aardvark

Group 3
zebra

Group 4
ant
duck

If no group label has been provided no header will be added.

The following hook is used just before the header information is appended:

\printunsrtglossarygrouphook{⟨internal cs⟩}

The argument is the internal command used to build the group header (which will then be appended to main internal command containing the glossary code). This hook may be redefined to insert any additional code before the heading. Use \pretoc#1{⟨content⟩} if you want to insert ⟨content⟩ before the header and \appto#1{⟨content⟩} if you want to insert ⟨content⟩ after the header. (You can reference the entry label with \glscurrententrylabel and the current hierarchical level with \glscurrententrylevel but make sure they are expanded if they occur in ⟨content⟩.) For example, \printunsrttable redefines this hook to finish off the current row before the group header is added.

The above document can be modified to show the sub-group headings:
8. Defining and Displaying Glossaries

\renewcommand*{\glsxtraddgroup}[2]{#2}
\printunsrtglossary
\renewcommand*{\glsxtraddgroup}[2]{\ifnum\glscurrententrylevel<2 #2\fi}
\printunsrtglossary[type=another]

The result is shown in Example 139.

Example 139: Displaying unsorted glossaries with custom groups and sub-group headings

Glossary

Group 1
waterfowl
ant
zebra
aardvark

Group 2
adder
gazelle

Group 4
duck
mallard

Another Glossary

mallard

Group 4
aardvark
ant

Group 3
zebra

Group 2
duck

Note that the mallard entry (which has hierarchical level 2) has its group shown in the first glossary (where the group is formed for all levels) but not in the second glossary (where the redefinition of \glsxtraddgroup restricts group formation to just level 0 and level 1).

There’s a small visual distinction between the group titles in different hierarchical levels in the above. The top-level (level 0) groups have the title centred, whereas the sub-groups have their titles indented by the same amount as the corresponding sub-entries. This is due to the glossary style. Other styles may use the same formatting for all hierarchical levels.

397
8. Defining and Displaying Glossaries

The glossary styles provided with glossaries-extra and the base styles patched by glossaries-extra-stylemods all redefine:

\glossarygroupheading{⟨previous level⟩}{⟨level⟩}{⟨parent-label⟩}{⟨group-label⟩}

to format the sub-group headings in a manner applicable to the style. For example, styles that don’t show sub-entry names typically redefine this command to do nothing.

A default definition that simply does \glossarygroupheading{⟨group-label⟩} is automatically initialised by \setglossarystyle (via \glsxtrpreglossarystyle) to allow for styles that don’t redefine this command. The first two arguments refer to the hierarchical level, where ⟨previous level⟩ is the level of the previous group and ⟨level⟩ is the level of this new sub-group. The ⟨parent-label⟩ is the label of the current entry’s parent, where the current entry is the first entry of the sub-group that immediately follows the heading.

The bookindex style defines \glossarygroupheading to use the style’s associated command \glsxtrbookindexformatsubheader. This can be redefined as required. For example, the following uses the parent entry’s hierarchical information:

\renewcommand*{\glsxtrbookindexformatsubheader}[5]{% 
\ifnum#2>1\relax
\glistreesubsubitem\glistreegroupheaderfmt{\GlsXtrhiername{#3}
/ #5}%
\else
\glistreesubitem\glistreegroupheaderfmt{\GlsXtrhiername{#3} / #5}%
\fi
}

The above examples are contrived and demonstrate the need to define entries in a sensible order to achieve a sensible glossary with \printunsrtglossary. If you want to use this approach to display a glossary, you would need to make sure that you take care with the order that you define entries. This can be quite tedious for a large number of entries.

To switch to bib2gls, the entry data needs to be provided in a bib file. For example, the file animalfamilies.bib might contain:

@index{waterfowl, user1={Anseriformes}}
@index{ant, user1={Formicidae}}
@index{adder, user1={Vipera berus}}
@index{duck, parent={waterfowl}, user1={Anatidae}}
@index{zebra, user1={Hippopotigris}}
@index{aardvark, user1={Orycteropus afer}}
@index{gazelle, user1={Gazella}}
I’ve included some additional information stored in the user1 field that wasn’t in the earlier examples. The document needs to use the record option and \GlsXtrLoadResources in order for it to work properly with bib2gls:

\usepackage[record,stylemods=bookindex,style=bookindex]{glossaries-extra}
\newglossary*{another}{Another Glossary}{animalfamilies}{en-GB}{la:user1:another}
\glsdefpostname{index}{\emph{\glsentryuseri{\glscurrententrylabel}}}\begin{document}
\printunsrtglossary\printunsrtglossary[type=another]\end{document}

If this code is saved in the file myDoc.tex then the build process is now:

pdfflatex myDoc
bib2gls --group myDoc
pdfflatex myDoc

The --group (or -g) switch is important as it instructs bib2gls to set the group field for the primary sort and the secondarygroup for the secondary sort. The primary sort will sort entries according to en-GB (British English). This can simply be set to en without a region. The secondary sort will resort the entries, but this time according to la (Latin) using the user1 key as the sort value. The entry labels will then be copied to the custom another glossary.

The glstex file created by bib2gls (which will then be input by \GlsXtrLoadResources on the subsequent \LaTeX run) essentially contains the following code:

\glsxtrsetgrouptitle{6881280}{W}
\glsxtrsetgrouptitle{5832704}{G}
\glsxtrsetgrouptitle{5373952}{A}
\glsxtrsetgrouptitle{7077888}{Z}
8. Defining and Displaying Glossaries

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>aardvark</td>
<td>Orycteropus afer, group={5373952}</td>
</tr>
<tr>
<td>adder</td>
<td>Vipera berus, group={5373952}</td>
</tr>
<tr>
<td>ant</td>
<td>Formicidae, group={5373952}</td>
</tr>
<tr>
<td>gazelle</td>
<td>Gazella, group={5832704}</td>
</tr>
<tr>
<td>waterfowl</td>
<td>Anseriformes, group={6881280}</td>
</tr>
<tr>
<td>duck</td>
<td>Anseriformes, group={6881280}</td>
</tr>
<tr>
<td>mallard</td>
<td>Anas platyrhynchos, group={}</td>
</tr>
<tr>
<td>zebra</td>
<td>Hippotigris, group={7077888}</td>
</tr>
</tbody>
</table>

It’s more complicated than this as helper commands are provided to make it easier to customize and the entries will all have category={index} since they were defined with
8. Defining and Displaying Glossaries

@index, but this is basically like the preamble in the earlier examples, except that the ordering and groups are more logical. The result is shown in Example 140.

Example 140: Displaying sorted glossaries with groups using \bib2gls

Glossary

\begin{tabular}{ll}
A & W \\
\hline
aardvark (\textit{Orycteropus afer}) & waterfowl (\textit{Anseriformes}) \\
adder (\textit{Vipera berus}) & duck (\textit{Anatidae}) \\
ant (\textit{Formicidae}) & mallard (\textit{Anas platyrhynchos}) \\
G & Z \\
gazelle (\textit{Gazella}) & zebra (\textit{Hippotigris}) \\
\end{tabular}

Another Glossary

\begin{tabular}{ll}
A & H \\
\hline
waterfowl (\textit{Anseriformes}) & zebra (\textit{Hippotigris}) \\
duck (\textit{Anatidae}) & \\
mallard (\textit{Anas platyrhynchos}) & \\
F & O \\
ant (\textit{Formicidae}) & \\
G & V \\
gazelle (\textit{Gazella}) & adder (\textit{Vipera berus}) \\
\end{tabular}

Note that the \texttt{group} and \texttt{secondarygroup} fields haven’t been set for the child entries (duck and mallard). This is the default behaviour and it means that regardless of the definition you provide for \glsxtraddgroup, sub-groups won’t be displayed. If you want those fields set for child entries, you need to use the \texttt{group-level} resource option. For example:

\begin{verbatim}
\GlsXtrLoadResources[selection=all,% select all entries
  group-level={<=1},% level 0 and 1
  src={animalfamilies},% identify bib file(s)
\end{verbatim}

401
8. Defining and Displaying Glossaries

```latex
\begin{Verbatim}
\texttt{sort=en-GB,}\% sort method
secondary={\texttt{la:user1:another}}\% sort again and copy
\end{Verbatim}
```

This will add support for level 0 (no parent) and level 1 (parent but no grandparent) entries. Deeper levels won’t have support. The `--group` switch is still required.

### 8.4.2. Location Lists

The “unsrt” family of commands check for the existence of the `location` and `loclist` keys. These are both defined by the `record` option. (The `loclist` field is also used by `\makeindex` but isn’t defined as a key.)

The `location` field (if set) should contain the formatted location list. This is checked first and used if not empty. Otherwise the `loclist` field (if set) is used, but that will use the same method as `\makeindex` but doesn’t compact consecutive locations.

It’s possible to choose a different field for the formatted location list by redefining:

```
\GlsXtrLocationField initial: location
```

This should expand to the internal field label. If the field is not the default `location` then the test for `loclist` is omitted.

Whichever field is used, the formatted location list is passed to the appropriate glossary style command (\glossentry or \subglossentry) encapsulated with \glossaryentrynumbers.

If there’s no location field or if the tested fields are empty, then an empty argument (with no `\glossaryentrynumbers` encapsulator) is passed to the glossary style command. In this case, the `nonumberlist` option is redundant as there’s no location list to suppress.

### 8.4.3. Advanced Commands

To provide a better understanding of how filtered and inner glossaries work, it’s useful to understand the difference between `\printglossary` (used with `makeindex` and `xindy`) and `\printunsrtglossary` (used with `bib2gls`).

In the first case, `makeindex` or `xindy` is used to create a file that contains content in the form:

```
\glossarysection{\texttt{\glossarytoctitle}}{\texttt{\glossarytitle}}%
\glossarypreamble
\begin{the glossary}\glossaryheader
\texttt{(content)}
\end{the glossary}\glossarypostamble
```

402
8. Defining and Displaying Glossaries

where \langle content \rangle contains lines such as:

\glsgroupheading{\langle group label \rangle}\relax \glsresetentrylist \glossentry{\langle entry label \rangle}{\langle location list \rangle}\
\subglossentry{\langle level \rangle}{\langle entry label \rangle}{\langle location list \rangle}\

The group headings (see §8.4.1) are typeset using \glsgroupheading. Top-level entries are typeset with \glossentry and child entries are typeset with \subglossentry where \langle level \rangle indicates the hierarchical level. Both makeindex and xindy order the items so that the child entries are placed immediately after the corresponding parent entry.

The \printglossary command essentially does:

\langle Set default title and style.\rangle
\bgroup
\langle Initial setup.\rangle
\langle Input the file created by makeindex or xindy.\rangle
\egroup

The initial setup part sets the glossary style (which determines the definitions of the glossary, \glossaryheader, \glsgroupheading, \glossentry and \subglossentry), assigns the title (\glossarytitle and \glossarytoctitle) and defines \currentglossary. (There is some other stuff done both before and after the file is input, but that’s not relevant here.)

In the case of bib2gls, there isn’t a glossary file to input. Instead, bib2gls is used to create a file that contains the entry definitions, which is input in the document preamble (via \GlsXtrLoadResources). The entries are defined in the required order and use internal fields to store the indexing information (such as the group label and location lists). Now \printunsrtglossary is used to display the glossary, which essentially does:

\langle Set default title and style.\rangle
\bgroup
\langle Initial setup.\rangle
\langle Construct internal control sequence containing glossary content.\rangle
\langle Expand internal control sequence.\rangle
\langle Expand internal control sequence.\rangle
\langle Expand internal control sequence.\rangle
\egroup

The \langle initial setup \rangle is the same as for \printglossary. The key difference here is that there’s no file containing the typeset glossary that can be simply input. Instead it’s necessary
to iterate over the glossary’s internal label list. Some of the glossary styles use a tabular-like environment (such as longtable, which is used by the long styles). It’s always problematic having a loop inside a tabular context so \printunsrtglossary by-passes the problem by moving the loop outside of the theglossary environment. The command iterates over all entry labels (in the order in which they were added to the glossary) and constructs an internal control sequence (@glsxtr@doglossary), which ends up containing:

\begin{theglossary}
glossaryheader
\begin{code}
\content
\end{code}
\end{theglossary}

Note that \glsresetentrylist has been removed in v1.50 since it’s generally unnecessary with bib2gls and causes interference with tabular styles.

The \begin{code} can be inserted just after \begin{theglossary} by the command:

\printunsrtglossarypostbegin{\internalcs}

This does nothing by default (so \begin{code} will be omitted). If you still need to have \glsresetentrylist at the start, you can redefine this hook as follows:

\renewcommand*{\printunsrtglossarypostbegin}{\appto#1{\glsresetentrylist}}%

The \end{code} can be inserted just before \end{theglossary} by the command:

\printunsrtglossarypreend{\internalcs}

This does nothing by default (so \end{code} will be omitted). (These two hooks are only used in \printunsrtglossary not in by \printunsrtinnerglossary or printunsrtglossarywrap.) For example, \printunsrttable redefines the end hook to finish off the final row.

In both hooks, the argument will be @glsxtr@doglossary and, in both cases, you need to use \appto within the definition in order to insert \begin{code} and \end{code} in the correct place. If you use \preto, the code will end up at the start, before \begin{theglossary}.
The \langle content \rangle in this case is different as it doesn’t explicitly contain \glossentry and \subglossentry but instead uses an internal handler that just takes the entry label as the argument. The \glsheading{\langle group label \rangle} command is inserted whenever a top-level entry has the group field set to a label that’s different to the previous top-level entry’s group field (and, if supported, sub-groups are similarly inserted with \glssubgroupheading, see §8.4.1). So the content is in the form:

\glsheading{\langle group label \rangle}\%
\langle internal cs handler \rangle{\langle entry label \rangle}\%
\langle internal cs handler \rangle{\langle entry label \rangle}\%
...
\glsheading{\langle group label \rangle}\%
\langle internal cs handler \rangle{\langle entry label \rangle}\%
\langle internal cs handler \rangle{\langle entry label \rangle}\%
...

There are hooks and commands available for use within those hooks that may be adjusted to customize the way the glossary is displayed. These are described below.

At each iteration (while the glossary content is being constructed), the following steps are performed:

1. Store the current entry label in \glscurrententrylabel.

2. If \glscurrententrylabel is empty, skip this iteration.

3. Define placeholder commands:

\glscurrententrylevel

This is set to the current entry’s hierarchical level (taking leveloffset and flatten options into account);

\glscurrenttoplevelentry

This is set to the current entry label if \glscurrententrylevel is 0 (that is, it expands to the most recent top-level entry, allowing for flatten and leveloffset);
8. Defining and Displaying Glossaries

This is set to the current entry label if `flatten=true` or if the current entry doesn’t have a parent (that is, it expands to the most recent top-level entry, allowing for `flatten` but not `leveloffset`).

4. Perform the entry process hook:

\printunsrtglossaryentryprocesshook\{\langle entry-label\rangle\}

This does nothing by default. Within the definition of this hook, you may use:

\printunsrtglossaryskipentry

This will cause the remainder of the current iteration to be skipped, which will prevent the current entry from being shown in the glossary.

5. If `groups=true`, use `\glsxtraddgroup` (see §8.4.1) to append the top-level group heading (`\glsgroupheading`) or the sub-group heading (`\glssubgroupheading`) to `\@glsxtr@doglossary`.

6. Perform the pre-entry process hook:

\printunsrtglossarypreentryprocesshook\{\langle internal cs\rangle\}

The argument will be `\@glsxtr@doglossary`. This may be used to insert any additional content before the entry (use `\appto#1\{\langle content\rangle\}`). (The entry label can be referenced with `\glscurrententrylabel` but make sure it’s expanded if it occurs in `\langle content\rangle`.) For example, `\printunsttable` redefines this hook to insert and \texttt{\tabularnewline} between blocks.

7. Append `\langle internal cs handler\rangle\{\langle entry label\rangle\}` to `\@glsxtr@doglossary`.

8. Perform the post-entry process hook:

\printunsrtglossarypostentryprocesshook\{\langle internal cs\rangle\}

The argument will be `\@glsxtr@doglossary`. This may be used to append any additional content after the entry (use `\appto#1\{\langle content\rangle\}`). (The entry label can be referenced with `\glscurrententrylabel` but make sure it’s expanded if it occurs in `\langle content\rangle`.) For example, `\printunsttable` redefines this hook to reset the block index if the end of a row has been reached.
The placeholders \glscurrenttoplevelentry and \glscurrentrootentry may not be an ancestor of the current entry. For example, if the glossary doesn’t have child entries immediately following their parent entry.

Once the glossary construction (\@glsxtr@doglossary) has been completed, the following hook is performed:

\printunsrtglossarypredoglossary

This does nothing by default. You can redefine this to show the definition of \@glsxtr@doglossary for debugging purposes:

\renewcommand{\printunsrtglossarypredoglossary}{\csshow{@glsxtr@doglossary}}

This will interrupt the \LaTeX run and display the definition in the transcript.

The handler command \langle internal cs handler \rangle performs the following:

\protected@xdef\glscurrententrylabel{⟨entry-label⟩}%
\printunsrtglossaryhandler\glscurrententrylabel

This stores the entry’s label in \glscurrententrylabel (which allows it to be referenced in style hooks, such as the post-name hook or post-description hook). Note that it uses a global definition to avoid scoping issues caused with tabular-like styles. The main handling of the entry is performed by:

\printunsrtglossaryhandler{⟨entry-label⟩}

This is simply defined to use:

\glsxtrunsrtdo{⟨entry-label⟩}

This displays the entry according to the current glossary style, taking the hierarchical level into account (as given by \glscurrententrylevel).
The following are additional commands that may be useful in the above hooks.

\glsxtriflabelinlist{⟨label⟩}{⟨label-list⟩}{⟨true⟩}{⟨false⟩}

Does ⟨true⟩ if the given label is in the given comma-separated list of labels, otherwise does ⟨false⟩. The label and list are fully expanded.

\ifglsxtrprintglossflatten ⟨true⟩\else ⟨false⟩\fi

initial: \iffalse

This conditional is set by the flatten option and can be used to test if the option has been set.

For example, the following skips all entries that have the category set to symbol:

\usepackage[style=index]{glossaries-extra}
\newglossaryentry{ant}{name={ant},description={}}
\newglossaryentry{pi}{name={\ensuremath{\pi}},description={},
category={symbol}}
\newglossaryentry{aardvark}{name={aardvark},description={}}
\newglossaryentry{alpha}{name={\ensuremath{\alpha}},description={},
category={symbol}}
\begin{document}
\renewcommand{\printunsrtglossaryentryprocesshook}[1]{%
\glsifcategory{#1}{symbol}%
{\printunsrtglossaryskipentry}%
{}%
}
\printunsrtglossary
\end{document}

Example 141: Filtering by category

Glossary

ant
aardvark
8. Defining and Displaying Glossaries

8.4.3.1. Inner Glossaries

See also Gallery: Inner or Nested Glossaries.

\[\text{dickimaw-books.com/gallery/index.php?label=bib2gls-inner}\]

It’s possible you may want to combine multiple glossaries sequentially, as sub-blocks of a single list. The inner part of \texttt{\textbackslash printunsrtglossary} can be created with:

\begin{verbatim}
\textbackslash printunsrtinnerglossary[\langle options\rangle]{\langle pre-code\rangle}{\langle post-code\rangle}
\end{verbatim}

This can’t be used on its own, as it only forms a fragment. It doesn’t include the section header, style initialisation, preamble, \texttt{theglossary} environment, header and postamble. As with \texttt{\textbackslash printunsrtglossary}, \texttt{\textbackslash printunsrtinnerglossary} constructs an internal control sequence containing the content, but it adds scoping to localise the effects of any supplied options. So it essentially does:

\begin{verbatim}
\begingroup
\langle Initial setup (process options).\rangle
\langle pre-code\rangle
\langle Construct internal control sequence containing glossary content.\rangle
\langle Expand internal control sequence.\rangle
\langle post-code\rangle
\endgroup
\end{verbatim}

There are two ways this command may be used.

\begin{verbatim}
\begin{printunsrtglossarywrap}[\langle options\rangle] \langle content\rangle \end{printunsrtglossarywrap}
\end{verbatim}

The \texttt{\textbackslash printunsrtglossarywrap} environment takes one optional argument that uses the same keys as \texttt{\textbackslash printunsrtglossary} (see §8.3). Note that in this case the \texttt{type} key simply provides a title (if one has been assigned to that glossary). It doesn’t indicate the content. There’s no point using both \texttt{type} and \texttt{title}.

The start of the environment sets up the glossary style and does the header:
8. Defining and Displaying Glossaries

\glossarysection\[\glossarytoctitle\]{\glossarytitle}\
\glossarypreamble
\begin{theglossary}\glossaryheader\glsresetentrylist
The end of this wrapper environment ends the glossary and does the postamble:
\end{theglossary}\glossarypostamble

Note that the \printunsrtglossarypostbegin, \printunsrtglossarypreend and \printunsrtglossarypredoglossary hooks aren't used.
For example, the following has two glossaries but displays them as inner glossaries:

\newglossaryentry{ant}{name={ant},description={}}
\newglossaryentry{bee}{name={bee},description={}}
\newglossaryentry*{other}
\newglossaryentry{duck}{name={duck},description={}}
\newglossaryentry{goose}{name={goose},description={}}
\begin{document}
\begin{printunsrtglossarywrap}[style=index]
\glstreeitem First Glossary
\printunsrtinnerglossary[leveloffset=1]{}{}
\glstreeitem Second Glossary
\printunsrtinnerglossary[type=other,leveloffset=1]{}{}
\end{printunsrtglossarywrap}
\end{document}

Example 142: Inner glossaries using printunsrtglossarywrap

Glossary

First Glossary
  ant
  bee
Second Glossary
  duck
  goose
The other way that `\printunsrtinnerglossary` can be used is within `\printunsrtglossary`. The handler function described in §8.4.3 that’s used to process each entry to be displayed in the glossary, is defined as:

\[\text{\texttt{\newcommand{\printunsrtglossaryhandler}[1]{\glsxtrunsrtdo{#1}}}}\]

It’s possible to redefine this so that it also displays an inner glossary. The following example has the terms “pictograph” and “Greek symbol” in the main glossary. Two ignored glossaries are created (which don’t require a title) where the glossary label matches an entry label in the main glossary.

\[\text{\texttt{\usepackage{fontawesome} \usepackage[style=tree]{glossaries-extra} \newglossaryentry{pictograph}{name={pictograph}, description={picture or symbol representing a word or phrase}} \newglossaryentry{mathgreek}{name={Greek symbol}, description={mathematical constants or functions}} \newignoredglossary*{pictograph} \newignoredglossary*{mathgreek} \newglossaryentry{cut}{type={pictograph}, name={\faCut}, description={cut}} \newglossaryentrypaste{type={pictograph}, name={\faPaste}, description={paste}} \newglossaryentry{alpha}{type={mathgreek}, name={\ensuremath{\alpha}}, description={alpha}} \newglossaryentry{beta}{type={mathgreek}, name={\ensuremath{\beta}}, description={beta}} \begin{document} \newcommand{\nestedhandler}[1]{% \glsxtrunsrtdo{#1}% \iff伪ositoryexists*{#1}% \% \printunsrtinnerglossary[type={#1},leveloffset={++1}, groups=false]{% }% \% \% \% \end{document}}\]

411
8. Defining and Displaying Glossaries

This creates a custom command \nestedhandler that can be used as the handler to create nested glossaries. After each item in the glossary, if the entry’s label matches the label of a defined glossary, that glossary is displayed with its hierarchical level incremented by 1, which creates the illusion of child entries. The resulting document is shown in Example 143.

Example 143: Nested glossaries

<table>
<thead>
<tr>
<th>Glossary</th>
</tr>
</thead>
<tbody>
<tr>
<td>pictograph</td>
</tr>
<tr>
<td>cut</td>
</tr>
<tr>
<td>Greek symbol</td>
</tr>
<tr>
<td>α</td>
</tr>
</tbody>
</table>

8.4.3.2. Per-Unit Glossaries

If you are using bib2gls then it’s possible to only list entries that match a particular counter value. For example, you may want a mini-glossary at the start of a section that only lists the entries that have been recorded in that section. This can be done by using the handler to skip entries that don’t have a matching record. It can also be implemented with record counting, as shown in Example 156 on page 558 in §11.4.

It’s also possible to make each indexing instance automatically make a note of a particular counter using:

\GlsXtrRecordCounter{⟨counter-name⟩}

(This doesn’t correspond to a bib2gls record. That’s dealt with by the indexing that comes first.)

This command may only be used in the preamble (with record) and indicates that whenever an entry is indexed, the following line should be added to the aux file:

\glsxtr@counterrecord{⟨entry-label⟩}{⟨counter⟩}{⟨value⟩}
where \( \langle value \rangle \) is given by \( \texttt{\textbackslash the} \langle counter \rangle \). On the next \LaTeX{} run, this information is picked up from the aux file and the information is added to the \texttt{record.\langle counter\rangle} field (stored as an etoolbox internal list). This internal command is only used in the aux file and has a user-level hook:

\[
\texttt{\textbackslash glsxtrAddCounterRecordHook\{\langle entry-label\rangle\}/\langle\langle counter\rangle\rangle/\langle\langle value\rangle\rangle}
\]

This does nothing by default. If you want to redefine this, the redefinition must be placed in the document preamble before the aux file is input.

There are two ways of skipping an entry. The first is to redefine \texttt{\textbackslash printunsrtglossary-entryprocesshook} to perform the test and use \texttt{\textbackslash printunsrtglossaryskipentry} to skip an unwanted entry (as illustrated earlier). The second is to perform the test in \texttt{\textbackslash printunsrtglossary-handler}. The first method is the better option for large lists that may contain group headers. The example below uses the second method.

The file \texttt{myentries.bib} contains the following:

\begin{verbatim}
@symbol{pi,name={\ensuremath{\pi}},
  description={ratio of the length of the circumference of a circle
to its diameter}}
@symbol{root2,name={\ensuremath{\sqrt{2}}},
  description={Pythagoras' constant}}
@symbol{zeta3,name={\ensuremath{\zeta(3)}},
  description={Ap\'ery's constant}}
@symbol{zero,name={0},
  description={nothing or nil}}
@symbol{one,name={1},
  description={single entity, unity}}
\end{verbatim}

The document redefines the handler to only show entries in the current section:

\begin{verbatim}
\usepackage[record,stylemods,style=index]{glossaries-extra}
\GlsXtrRecordCounter{section}
\GlsXtrLoadResources[src={myentries}]
\begin{document}
\renewcommand{\printunsrtglossaryhandler}[1][1]{%
  \glsxtrfieldxifinlist{#1}{record.section}{\thesection}
  {\glsxtrunsrtdo{#1}}%
  {}}%
\section{Sample}
\end{document}
\end{verbatim}
8. Defining and Displaying Glossaries

\printunsrtglos
tary
This section discusses \gls{pi}, \gls{root2} and \gls{zeta3}.

\section{Another Sample}
\printunsrtglos
tary
This section discusses \gls{one}, \gls{pi} and \gls{zero}.
\end{document}

If the document is saved in the file myDoc.tex then the build process is:

pdflatex myDoc
bib2gls myDoc
pdflatex myDoc

The first \LaTeX run adds the records to the aux file for bib2gls to pick up, but also adds the \glsxtr@counterrecord lines (which bib2gls ignores) that setup the record.section list field for the given entry.

This means that \glsxtrfieldxifinlist can be used to determine whether or not the current section number (\thesection) is in the list. If it is, then the entry is displayed in the current glossary style using the default \glsxtrunsrtdo. Otherwise nothing is displayed.

The following command is provided that performs something similar:

\printunsrtglossaryunit [(options)]{(counter-name)}

This is equivalent to:

\printunsrtglossary* [type=\glsdefaulttype,#1]{%
  \printunsrtglossaryunitsetup{#2}%
}\

This initialises the hook via:

\printunsrtglossaryunitsetup{(counter-name)}

This is essentially does:
8. Defining and Displaying Glossaries

\% redefine handler to only show entries with a match:
\renewcommand{\printunsrtglossaryhandler}{\%
  \glsxtrfieldxifinlist{#1}{ record.⟨counter-name⟩ }{\the⟨counter-name⟩}%
  {\glsxtrunsrtdo{#1}}%
}%
\% (assign target name prefixes (see below))
\% suppress section header:
\renewcommand*{\glossarysection}{\%
  \% append vertical space after the glossary:
  \appto\glossarypostamble{\printunsrtglossaryunitpostskip}%
}\printunsrtglossaryunitpostskip

This is more complicated than the original example as it also suppresses the glossary section header and modifies the target name prefix. Additionally, the following is appended to the end of the glossary:

This simply does:
\glspar\medskip\glspar

which creates a small vertical space. The target name prefix (\texttt{targetnameprefix}) is assigned as follows. If \texttt{\the⟨counter-name⟩} has been defined, the prefix is:
record.⟨counter-name⟩.\the⟨counter-name⟩.\@gobble

otherwise the prefix is:
record.⟨counter-name⟩.\the⟨counter-name⟩.\@gobble

The use of \texttt{\@gobble} at the end discards \texttt{\gloolinkprefix}.

The above example can be rewritten using \texttt{\printunsrtglossaryunit}. I’ve added \texttt{symbol-sort-fallback} to sort by the description and a full glossary at the end of the document.
\begin{document}
\section{Sample}
\printunsrtglossaryunit{section}
This section discusses \gls{pi}, \gls{root2} and \gls{zeta3}.

\section{Another Sample}
\printunsrtglossaryunit{section}
This section discusses \gls{one}, \gls{pi} and \gls{zero}.

\printunsrtglossaries
\end{document}

The build process is the same as before:

\pdflatex myDoc  
\bib2gls myDoc  
\pdflatex myDoc

The resulting document is shown in Example 144 on the following page. Note that all glossaries show the location lists, which all contain the page number 1, since the example document is only one page long.

Other variations include creating a secondary glossary that’s ordered differently for the mini-glossaries. For example:

\newignoredglossary*{glossary2}
\GlsXtrLoadResources[\src={\jobname},
  \symbol-sort-fallback=description,
  \secondary=use:glossary2]

This orders the secondary glossary according to use (the first record for the entire document not for the given unit). The mini-glossaries will then need the \texttt{type} option:

\printunsrtglossaryunit[type=glossary2]{section}

There is an alternative method that ensures the mini-glossaries are ordered by use within the section. This can be done by redefining \texttt{\glsxtrAddCounterRecordHook} to create a glossary for each unit (instead of using a secondary glossary):
Example 144: Sub-glossary for a given counter value

1 Sample

ζ(3) Apéry’s constant
√2 Pythagoras’ constant
π ratio of the length of the circumference of a circle to its diameter

This section discusses π, √2 and ζ(3).

2 Another Sample

0 nothing or nil
π ratio of the length of the circumference of a circle to its diameter
1 single entity, unity

This section discusses 1, π and 0.

Glossary

ζ(3) Apéry’s constant
0 nothing or nil
√2 Pythagoras’ constant
π ratio of the length of the circumference of a circle to its diameter
1 single entity, unity

\renewcommand{\glxsxtrAddCounterRecordHook}[3]{% \\
prowdeignedglossary{#2.#3} \% \\
glsxtrcopytoglossary*{#1}{#2.#3} \% } 

(Remember this needs to be done in the preamble, before the aux file is input.)

This creates a glossary with the label ⟨counter⟩.⟨value⟩, if it’s not already defined, and adds the entry’s label to it. This means that this glossary will only contain the entries for the matching ⟨counter⟩ and ⟨value⟩, and the entry labels are in the order they were added to the aux file.

The glossary needs to be set appropriately. For example:
8. Defining and Displaying Glossaries

There’s now no filtering required, but \texttt{\printunsrtglossaryunit} is still useful as it automatically suppresses the section header, alters the hyperlink prefix and adds extra spacing after the glossary. However, if you prefer, you can simply do something like:

\begin{verbatim}
\printunsrtglossary*[type=section.,thesection, target=false]
\renewcommand*{\glossarysection}[2][]{}
\printunsrtglossaryunitpostskip
\end{verbatim}

This is done in Example 145.

Example 145: Sub-glossary for a given counter value ordered by use in the section

\begin{enumerate}
\item Sample

\begin{itemize}
\item $\pi$ ratio of the length of the circumference of a circle to its diameter
\item $\sqrt{2}$ Pythagoras’ constant
\item $\zeta(3)$ Apéry’s constant
\end{itemize}

This section discusses $\pi$, $\sqrt{2}$ and $\zeta(3)$.

\item Another Sample

\begin{itemize}
\item 1 single entity, unity
\item $\pi$ ratio of the length of the circumference of a circle to its diameter
\item 0 nothing or nil
\end{itemize}

This section discusses 1, $\pi$ and 0.

\end{enumerate}

Glossary

\begin{itemize}
\item $\zeta(3)$ Apéry’s constant
\item 0 nothing or nil
\item $\sqrt{2}$ Pythagoras’ constant
\item $\pi$ ratio of the length of the circumference of a circle to its diameter
\item 1 single entity, unity
\end{itemize}
8.5. Standalone Entry Items

It may be that you don’t want a list but would rather display entry details throughout the document. You can simply do \glsentryname followed by \glsentrydesc. (Remember that if you don’t want a sorted list, use sort=none or sort=clear to skip the preprocessing of the sort field.) For example, in the preamble provide a custom command to display the entry’s name and description:

\newcommand{\displayterm}[1]{%
  \par\medskip
  \noindent
  Definition: \glsentryname{#1}.\par
  \glsentrydesc{#1}
  \par\medskip
}\%

define your entries, for example:

\newglossaryentry{function}{name={function},
  description={a relation or expression involving variables}}

and then later in the text:

\displayterm{function}

However, if may be that you want to use hyperref and have commands like \gls link back to the place where the term is described. Instead of using \glsentryname use:

\glsxtrglossentry{⟨entry-label⟩}

where ⟨entry-label⟩ is the entry’s label.

This is designed to behave much like the way the name is displayed in the glossary. It performs the following:

- Defines \glscurrententrylabel to the entry’s label. This is usually done at the start of the glossary style commands \glossentry and \subglossentry and may be used by hooks, such as the post-name hook. Here the definition is localised so that it’s only available for use in \glsentryname.

- Defines \currentglossary to the entry’s glossary type. This is usually done at the start of commands like \printglossary and may be used by style hooks. Here the
8. Defining and Displaying Glossaries

definition is localised so that it’s only available for use in \glsentryitem and \gls-
subentryitem. The value is obtained by fully expanding:

\GlsXtrStandaloneGlossaryType

which defaults to the value of the type field for the current entry.

• Increments and display the entry counters if the entrycounter or subentrycounter
package options are set. If the entry doesn’t have a parent, then this does:

\glsentryitem{⟨entry-label⟩}

otherwise it does:

\GlsXtrStandaloneSubEntryItem{⟨entry-label⟩}

which defaults to \glssubentryitem{⟨entry-label⟩} if the entry has a parent but not
a grandparent.

This reflects the behaviour of the predefined hierarchical styles. A bug in pre-version 1.31
used \glsentryitem for all child levels, which doesn’t match the hierarchical glossary
styles. If you want to restore this behaviour, just do:

\renewcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
  \glssubentryitem[#1]}

• Sets the hyper-target if supported (using \glstarget) and displays the entry name
using:

\GlsXtrStandaloneEntryName{⟨entry-label⟩}

which uses \glstarget{⟨entry-label⟩}{\glossentryname{⟨entry-label⟩}} by default.
Remember that \glossentryname uses \glsnamefont or picks up the style from cat-
egory attributes such as glossnamefont. This can result in duplicate targets if you use
both standalone commands and display the glossary. In which case, you can redefine
\glstarget to use \glsxtrtarget, which will ensure that the first target will be the
one that takes precedence.
If you have used \nopostdesc or \glsxtrnopostpunc in any of your description fields, you can use:

\glsxtractivatenopost

to make these commands behave as they normally do within a glossary. This needs to be placed before:

\glossentrydesc\{\langle entry-label \rangle\}\glspostdescription

and scoped. Note that \glsnonextpages and \glsnextpages have no effect outside of the glossary and are not intended for use in a standalone context.

It’s also possible to select a different field (rather than using \textit{name}):

\glsxtrglossentryother\{\langle header \rangle\}\{\langle entry-label \rangle\}\{\langle field-label \rangle\}

The \langle field-label \rangle must be given using its internal field label. The \langle header \rangle argument is the code to pass to the third argument of \glsxtrtitleorpdforheading. It may be left empty in which case the default is determined as follows:

- If \glsxtr\langle field-label \rangle is defined (see §5.3.3), then \langle header \rangle is \glsxtr\langle field-label \rangle\{entry-label\}.

- Otherwise \langle header \rangle is simply the field value.

The \glsxtrglossentryother command internally uses

\GlsXtrStandaloneEntryOther\{\langle entry-label \rangle\}\{\langle field-label \rangle\}

instead of \GlsXtrStandaloneEntryName, which uses \glossentrynameother\{\langle entry-label \rangle\}\{\langle field-label \rangle\} instead of \glossentryname\{\langle entry-label \rangle\}.

If you have loaded the glossaries-accsupp package (through the accsupp option) then accessibility support will be provided if there’s a corresponding command:

\gls\{\langle field-label \rangle\}accessdisplay\{\langle text \rangle\}\{\langle entry-label \rangle\}

(for example, \glssymbolaccessdisplay).

This means that my custom command can be changed to:

\newcommand{\displayterm}[1]{%\par\medskip\par\noindent

\gls{\langle field-label \rangle}\accessdisplay\{\langle text \rangle\}\{\langle entry-label \rangle\}

421
If I want numbered definitions, then I can use the package options `entrycounter` or `subentrycounter` and remove the colon:

\input{8. Defining and Displaying Glossaries}

\newcommand{\displayterm}[1]{%
  \par\medskip
  \noindent
  Definition \glsxtrglossentry{#1}.\par
  \glsentrydesc{#1}
  \par\medskip
}\%

The counter label uses a dot after the number by default but this can be changed to a colon:

\input{8. Defining and Displaying Glossaries}

\renewcommand*{\glsentrycounterlabel}{\theglossaryentry:space}

It's now possible to not only use \gls to link back to the definition but also use \glsrefentry to reference the counter and \glsxtrpageref to reference the page number.

If I want the description to behave more like it does in a glossary in need to make the following modification:

\input{8. Defining and Displaying Glossaries}

(Nota the grouping to localise \glsxtractivatenopost.)

You can also use \glsxtrglossentry within section headings. For example:
8. Defining and Displaying Glossaries

If \glsxtrglossentry occurs in a section title and hyperref has been loaded, then \glsxtrglossentry will expand in the PDF bookmark as:

\GlsXtrStandaloneEntryPdfName{⟨entry-label⟩}

This defaults to \glsentryname{⟨entry-label⟩} The page headers and table of contents will use

\GlsXtrStandaloneEntryHeadName{⟨entry-label⟩}

which defaults to \glsxtrheadname{⟨entry-label⟩}. For example, to ensure that the name is displayed in sentence case in the title, PDF bookmarks and heading:

\glssetcategoryattribute{general}{glossname}{firstuc}
\renewcommand{\GlsXtrStandaloneEntryPdfName}[1]{\Glsentryname{#1}}
\renewcommand{\GlsXtrStandaloneEntryHeadName}[1]{\Glsentryname{#1}}

Note that this requires glossaries v4.50+ to ensure that \glsentryname expands. An alternative is to use \glsxtrusefield.

If \glsxtrglossentryother occurs in a section title and hyperref has been loaded, then \glsxtrglossentryother will expand in the PDF bookmark as:

\GlsXtrStandaloneEntryPdfOther{⟨entry-label⟩}{⟨field-label⟩}

This defaults to the value of the given field. The page headers and table of contents will use the ⟨header⟩ argument, if not empty, otherwise it will use:

\GlsXtrStandaloneEntryHeadOther{⟨entry-label⟩}{⟨field-label⟩}

This does \glsxtrhead{field-label}, if it exists, or otherwise it just does the value of the given field (which can be obtained with \glsxtrusefield).

If you're using a page style or table of contents that doesn't use \markright or \markboth or @starttoc then you need to insert \glsxtrmarkhook and \@glsxtrinmark at the start of the header or table of contents either scoped or afterwards cancelled with \@glsxtrnot-inmark and \glsxtrrestoremarkhook, see §5.3.3.
8. Defining and Displaying Glossaries

8.6. Glossary Style Modifications

The glossaries-extra package redefines \setglossarystyle, and it now includes a hook that’s performed before the style is set:

\glsxtrpreglossarystyle

This allows for new style commands that aren’t provided by the base glossaries package to be initialised in the event that a style that doesn’t redefine them is used. The default definition is:

\newcommand{\glsxtrpreglossarystyle}{%
\renewcommand{\glssubgroupheading}[4]{\glsgroupheading{##4}}%
}

If you prefer a different default, you can redefine this command as appropriate.

The commands \glossentryname and \glossentrydesc are modified to take into account the glossname, glossnamefont, glossdesc and glossdescfont attributes (see §10). This means you can make simple font or case-changing modifications to the name and description without defining a new glossary style.

The command \glossentrysymbol is modified to take into account the glosssymbol-font attribute. Note that, unlike the above, there’s no corresponding attribute to change the case as it’s usually not appropriate to change the case of a symbol (and for some symbols, such as pictographs, there’s no concept of case). If \texorpdfstring has been defined \glossentrysymbol will be defined to do:

\texorpdfstring{⟨TEX code⟩}{⟨PDF⟩}

The ⟨TEX code⟩ part is robust and deals with the actual typesetting of the symbol. The ⟨PDF⟩ part is simply:

\glsentrypdfsymbol{⟨entry-label⟩}

which is defined to just do \glsentrysymbol{⟨entry-label⟩}. The chances are that the code in the symbol key won’t be valid in the PDF bookmarks, so you can redefine \glsentrypdfsymbol to use a more appropriate field. (If you do redefine this command, remember that it needs to fully expand.)

For example, if you are using glossaries-accsupp, you could use the symbolaccess field:
8. Defining and Displaying Glossaries

\renewcommand{\glsentrypdfsymbol}{% \glsentrysymbolaccess{#1}}

Alternatively, if you are using \texttt{bib2gls} you can use the \TeX\ parser library to interpret a copy of the \texttt{symbol} field and use that. For example, with the resource options:

\renewcommand{\glsentrypdfsymbol}{% \glsentryuseri{#1}}

This copies the value of the \texttt{symbol} field to the \texttt{user1} field (\texttt{replicate-fields}) and then replaces the value of the \texttt{user1} field with its interpreted value (\texttt{interpret-fields}).

This means you can then do:

\renewcommand{\glsentrypdfsymbol}{% \glsentryuseri{#1}}

(You may need \texttt{X\LaTeX} or \texttt{Lua\LaTeX} with this method.) This allows \texttt{glossentrysymbol} to be used in a section heading with standalone definitions. See the \texttt{bib2gls} manual for further details about the \TeX\ interpreter.

If you want to adapt a style to use another field instead of \texttt{name}, you can use:

\glossentrynameother{⟨entry-label⟩}{⟨field-label⟩}

This behaves just like \texttt{glossentryname} (that is, it obeys the \texttt{glossname} attribute, uses either the \texttt{glossnamefont} attribute or \texttt{glsnamefont} to format the text, and uses the post-name hook) but the text is obtained from the field given ⟨\texttt{field-label}⟩ instead of \texttt{name}. The ⟨\texttt{field-label}⟩ argument must be the internal field label (for example desc rather than description).

8.6.1. Post-Name Hooks

The glossaries-extra package adds a hook to \texttt{glossentryname} and \texttt{Glossentryname} (which is used in glossary styles to display the entry’s name):

\glsxtrpostnamehook{⟨entry-label⟩}

This is the main post-name hook, which implements additional hooks to allow for customisation. By default, \texttt{glsxtrpostnamehook} checks the \texttt{indexname} attribute. If the attribute exists for the category to which the entry belongs, then the name is automatically indexed using:
8. Defining and Displaying Glossaries

\glsxtrdoautoindexname{(entry-label)}{indexname}

See §12 for further details.

The post-name hook \glsxtrpostnamehook will also use:

\glsxtrpostname{category}

if it exists. You can use \glscurrententrylabel to obtain the entry label with the definition of this command. For example, suppose you are using a glossary style the doesn’t display the symbol, you can insert the symbol after the name for a particular category, say, the “symbol” category:

\newcommand*{\glsxtrpostnamesymbol}{\space (\glsentrysymbol{\glscurrententrylabel})}

For convenience, you can use:

\glsdefpostname{(category)}{(definition)}

This is simply a shortcut for:

\csdef{\glsxtrpostname(category)}{(definition)}

Note that it doesn’t check if the command has already been defined.

The post-name hook also does:

\glsextrapostnamehook{(entry-label)}

(before \glsxtrpostname{category}) to allow for additional non-category related code. This does nothing by default.

8.6.2. Post-Description Hooks

The glossaries package provides the hook \glspostdescription, which is placed after the description in some of the predefined styles. The glossaries-extra-stylemods package modifies the predefined styles to ensure that they all use this hook. This provides a convenient way to make slight adjustments, such as appending content after the description, without having to define a custom glossary style.
8. Defining and Displaying Glossaries

The glossaries-extra package redefines \glspostdescription so that it includes the following hook:

\glsxtrpostdescription

This new hook simply performs the category post-description hook:

\newcommand*{\glsxtrpostdescription}{\csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}}%

The punctuation that is automatically inserted with postdot or postpunc is placed after \glsxtrpostdescription, not before.

If you want to modify the hook for all entries (without affecting the postpunc or postdot options), then redefine \glsxtrpostdescription. If you want to adjust this hook according to the entry’s category, then you can simply redefine the category post-description hook.

\glsxtrpostdesc{category}

Some common category post-description hooks are provided:

\glsxtrpostdescgeneral \hspace{1cm} initial: empty

The post-description hook for the general category.

\glsxtrpostdescterm \hspace{1cm} initial: empty

The post-description hook for the term category.

\glsxtrpostdescacronym \hspace{1cm} initial: empty

The post-description hook for the acronym category.
8. Defining and Displaying Glossaries

\glsxtrpostdescabbreviation

The post-description hook for the \texttt{abbreviation} category.

The above all do nothing by default. You can redefine them with \texttt{\renewcommand} or use:

\glsdefpostdesc\{\langle\text{category}\rangle\}\{\langle\text{definition}\rangle\}

This will define (or redefine) \texttt{\glsxtrpostdesc\{category\}}. The package options \texttt{symbols}, \texttt{numbers} and \texttt{index} provide corresponding category post-description hooks.

You can reference the current entry within these hooks using \texttt{\glscurrententrylabel}, which is defined within the glossary (any of the \texttt{\print...glossary} commands) and also within the standalone commands, such as \texttt{\glsxtrglossentry}.

\glsxtrnopostpunc

Suppresses the post-description punctuation that is automatically inserted by package options \texttt{postdot} or \texttt{postpunc}.

The glossaries package provides \texttt{\nopostdesc}, which may be used in the \texttt{description} to suppress the post-description hook for that entry. This suppresses both the post-description punctuation and the additional \texttt{\glsxtrpostdescription} hook. If you only want to suppress to punctuation, then use \texttt{\glsxtrnopostpunc} instead.

The post-description hooks are implemented by \texttt{\glspostdescription} within the glossary style. If this command isn’t used in the style, then the additional hooks won’t be available.

\glsxtrrestorepostpunc

If this command is placed in the definition of \texttt{\glsxtrpostdescription} or added to the category post-link hook, then it will counter-act any use of \texttt{\glsxtrnopostpunc} to restore the post-description punctuation.

These commands have no effect outside of the glossary (except with standalone entries that use \texttt{\glsxtractionenopost} and \texttt{\glspostdescription}, see §8.5).
8. Defining and Displaying Glossaries

8.6.3. Number (Location) List

The location list is now placed inside the argument of:

\GlsXtrFormatLocationList{⟨location list⟩}

This is internally used by \glossaryentrynumbers. The nonumberlist option redefines \glossaryentrynumbers so that it doesn’t display the number list, but it still saves the number list in case it’s required. The desired font formatting for the location list can now more easily be set by redefining \GlsXtrFormatLocationList, without interfering with \glossaryentrynumbers.

If you want to suppress the number list always use the nonumberlist option instead of redefining \glossaryentrynumbers to do nothing.

Note that if you are using the “unsrt” family of commands the location list will only be present if the appropriate field has been set (see §8.4.2). There’s no need to save locations with bib2gls or with \printnoidxglossary because this is performed automatically (unlike \printglossary where the trick with \glossaryentrynumbers is required to capture the location list).

Sometimes users like to insert “page” or “pages” in front of the location list. This is quite fiddly to do with the base glossaries package, but glossaries-extra provides a way of doing this. First you need to enable this option and specify the text to display using:

\GlsXtrEnablePreLocationTag{⟨page tag⟩}{⟨pages tag⟩}

where ⟨page tag⟩ is the text to display if the location list only contains a single location and ⟨pages tag⟩ is the text to display otherwise. For example:

\GlsXtrEnablePreLocationTag{Page: }{Pages: }

An extra run is required when using this command.

Use glsignore not @gobble as the format if you want to suppress the page number (and only index the entry once).

See the accompanying sample file sample-pages.tex for an example.
Note that \texttt{bib2gls} the \texttt{loc-prefix} resource option inserts a prefix at the start of non-empty location lists, which can be used as an alternative to \texttt{GlsXtrEnablePreLocationTag}. There is also a corresponding \texttt{loc-suffix} option to provide a suffix.

Location lists displayed with \texttt{printnoidxglossary} internally use: \texttt{\glsnoidxdisplayloc} This command is provided by \texttt{glossaries}, but is modified by \texttt{glossaries-extra} to check for the start and end range formation identifiers \( ( \text{and} ) \) which are discarded to obtain the actual control sequence name that forms the location formatting command.

If the range identifiers aren’t present, this just uses

\begin{verbatim}
\glsxtrdisplaysingleloc{⟨format⟩}{⟨location⟩}
\end{verbatim}

otherwise it uses

\begin{verbatim}
\glsxtrdisplaystartloc{⟨format⟩}{⟨location⟩}
\end{verbatim}

for the start of a range (where the identifier has been stripped from \texttt{⟨format⟩}) or

\begin{verbatim}
\glsxtrdisplayendloc{⟨format⟩}{⟨location⟩}
\end{verbatim}

for the end of a range (where the identifier has been stripped from \texttt{⟨format⟩}).

By default the start range command saves the format in:

\begin{verbatim}
\glsxtrlocrangefmt
\end{verbatim}

and does:

\begin{verbatim}
\glsxtrdisplaysingleloc{⟨format⟩}{⟨location⟩}
\end{verbatim}

(If the format is empty, it will be replaced with \texttt{glsnumberformat}.)

The end command checks that the format matches the start of the range, does:

\begin{verbatim}
\glsxtrdisplayendlochook
\end{verbatim}

(which does nothing by default), followed by:

\begin{verbatim}
\glsxtrdisplaysingleloc{⟨format⟩}{⟨location⟩}
\end{verbatim}
and then sets \texttt{\textglsxtrlocrangefmt} to empty.

This means that the list

\begin{verbatim}
\textbf{1},
\textbf{1},
\textbf{1}.
\end{verbatim}

doesn’t display any differently from

\begin{verbatim}
\textbf{1},
\textbf{1},
\textbf{1}.
\end{verbatim}

but it does make it easier to define your own custom list handler that can accommodate the ranges.

### 8.6.4. Indexing Groups

The letter or symbol or number groups are a by-product of the indexing application. These are usually determined during the sorting according to the first (significant) character of the sort value. If the first character is an alphabetical character, the group is a letter group, with the group label the same as the letter. If the sort value is numeric, the group is a number group, with the label \texttt{glsnumbers}, otherwise the group is a symbol group with the label \texttt{glssymbols}.

For the “unsrt” family of commands, see §8.4.1 for more details about how group headers are inserted into the glossary. Only those commands are able to support sub-groups.

With \texttt{xindy}, the number group is automatically provided with the \texttt{xindy=glsnumbers} package option. It can be suppressed with \texttt{xindy={glsnumbers=false}} (see the base glossaries user manual for further details).

With \texttt{bib2gls}, group formation requires --group (or -g). This setting is off by default to allow for a faster process where no groups are required. When this setting is on, there are additional groups, depending on the sort method. For example, if you use a date-time sort method, then you will have date-time groups.

Take care not to confusion groups with hierarchy. See Gallery: Logical Glossary Divisions (type vs group vs parent) \footnote{for the difference between the group, type and parent hierarchies} for the difference between the group, type and parent hierarchies.
The base glossaries package provides a simplistic way of assigning a title to a group to allow for the use of language-sensitive commands \texttt{\textbackslash glsymbols\textbackslash groupname} and \texttt{\textbackslash glsnumbers\textbackslash groupname}, which correspond to the \texttt{\textbackslash glssymbols} and \texttt{\textbackslash glsnumbers} groups. The more flexible groups that can be created with \texttt{\textbackslash bib2gls} require a better approach that is less likely to cause a conflict.

\begin{verbatim}
\glsxtrsetgrouptitle{\textbackslash group-label}{\textbackslash group-title}
\end{verbatim}

Globally assigns the given title \texttt{\textbackslash group-title} to the group identified by \texttt{\textbackslash group-label}. This command is used implicitly within the \texttt{glstex} file to assign titles to groups obtained by \texttt{\textbackslash bib2gls}. Judicious definitions of the helper commands provided by \texttt{\textbackslash bib2gls} can provide a more flexible way of assigning groups.

\begin{verbatim}
\glsxtrlocalsetgrouptitle{\textbackslash group-label}{\textbackslash group-title}
\end{verbatim}

As above but the assignment is local.

\begin{verbatim}
\glsxtrgrouptitle{\textbackslash group-label}{\texttt{	extbackslash cs}}
\end{verbatim}

Obtains the title corresponding to the group identified by \texttt{\textbackslash group-label} and stores the result in \texttt{\textbackslash cs}. This command first checks if a title has been assigned by \texttt{\glsxtrgrouptitle} and then, for compatibility with the base glossaries package, it will test for the existence of \texttt{\textbackslash group-label\textbackslash groupname} if \texttt{\textbackslash group-label} is \texttt{\textbackslash glssymbols} or \texttt{\textbackslash glsnumbers} or a single character. If no title is obtained from any of these tests, then the title will be assumed to be the same as the label.

The \texttt{\textbackslash printnoidxglossary} command has a slightly different method, which uses the character code so it’s not suitable with UTF-8. In general, \texttt{\textbackslash printnoidxglossary} is best avoided, where possible, and is inappropriate for locale-sensitive sorting.

\subsection*{8.6.5. glossaries-extra-stylemods}

The glossaries-extra-stylemods package (more conveniently loaded through the glossaries-extra stylemods option) modifies some of the predefined styles that are provided with the glossaries package.
Any styles loaded after glossaries-extra-stylemods won’t be patched.

The `stylemods` option may be provided without a value, in which case all currently defined styles will be patched. Alternatively, you can supply a comma-separated list as the value, which indicates that, for each `<element>` in the list, the package `glossary-<element>` should be loaded and, if it’s a package provided with the base `glossaries` package, patched. For example:

\begin{verbatim}
\usepackage{glossaries-extra}
\usepackage{glossary-longragged}
\usepackage{glossary-mcols}
\usepackage{glossaries-extra-stylemods}
\setglossarystyle{mcolindex}
\end{verbatim}

is equivalent to:

\begin{verbatim}
\usepackage[stylemods={longragged,mcols},style=mcolindex]{glossaries-extra}
\end{verbatim}

You may prefer to combine `stylemods=with nostyles` to reduce the overhead of loading unnecessary packages.

The `glossaries-extra-stylemods` package adjusts the predefined styles so that they all use `\glspostdescription` and replaces any hard-coded space before the location list with `\glsxtrprelocation initial: \space`

You can therefore redefine that command in combination with `postpunc` to alter the separator before the location list. For example, to have a comma followed by `\hfil`:

\begin{verbatim}
\usepackage[postpunc=comma,stylemods]{glossaries-extra}
\renewcommand{\glsxtrprelocation}{\hfil}
\end{verbatim}

Be careful with doing this as it will look odd if the location list is missing.

With `bib2gls` you can instead redefine `\glsxtrprelocation` to do nothing and set the location prefixes with `loc-prefix` which will only apply if the entry has a location list. Alternatively, you could redefine `\glsxtrprelocation` to check if the `location` field is set.
8. Defining and Displaying Glossaries

8.6.5.1. Inline Style

The patched inline style is dealt with slightly differently. The original definition provided by the glossary-inline package uses \glspostdescription at the end of the glossary (not after each entry description) within the definition of \glspostinline. The style modification changes this so that \glspostinline just does a full stop followed by space factor adjustment, and the description \glsinlinedescformat and sub-entry description formats \glsinlinesubdescformat are redefined to include \glssxtrpostdescription (not \glspostdescription). This means that the modified inline style isn't affected by the nopostdot option, but the category post-description hook can still be used.

8.6.5.2. Tabular Styles

The tabular-like styles, such as long are adjusted so that the \ifglsnogroupskip conditional (set with nogroupskip) is moved outside of the definition of \glsgroups to avoid problems that cause an “Incomplete \iftrue” error with \printunsrtglossary and \printnoidxglossary. This means that if you want to change this conditional using \setupglossaries or using the nogroupskip option in \printglossary, \printnoidxglossary or \printunsrtglossary, you must also reset the glossary style.

8.6.5.3. List Styles

The list styles use:

\glslistprelocation \textit{initial:} \glssxtrprelocation

(which defaults to \glssxtrprelocation) for top-level items and:

\glslistchildprelocation \textit{initial:} \glsslistprelocation

(which defaults to \glsslistprelocation) for child items.

The description (including the post-description hook) is governed by:

\glslistdesc\{\textit{entry-label}\}

for the list and altlist styles (but not the listdotted variations).

The hard-coded \texttt{\item[\texttt{(target and name)}]} is replaced with:

\glslistitem\{\textit{entry-label}\}
8. Defining and Displaying Glossaries

The altlist styles use:

\glsaltlistitem{⟨entry-label⟩}

which internally uses \glslistitem. The header item (for the list styles that should the group title, such as listgroup) is governed by:

\glslistgroupheaderitem{⟨group-label⟩}{⟨header code⟩}

This ignores the ⟨group-label⟩ by default and simply places the second argument in the optional argument of \item. The ⟨header code⟩ is the formatted group title, possibly including a hypertarget. The spacing after the group item is given by:

\glslistgroupafterheader

For just the list style and its letter group variations (not the altlist or listdotted variations) the location list for child entries is followed by:

\glslistchildpostlocation

which defaults to a full stop.

The default value of \glslistdottedwidth is changed so that it’s set at the start of the document (if it hasn’t been changed in the preamble). This should take into account situations where \hsize isn’t set until the start of the document.

The separator between groups (if not nogroupskip) is now given by:

\glslistgroupskip

This defaults to \indexspace with penalties to deter page breaks. This command isn’t used if nogroupskip is set.

8.6.5.4. Tree Styles

The group headings for styles like treegroup are formatted with:

\glstreegroupheaderfmt{⟨text⟩}
The navigation elements for styles like treehypergroup is formatted with:

\glstreenavigationfmt{(text)}

The above two commands are defined in terms of \glstreenamefmt, since that was the command originally used for the group headings and navigation. This now allows these different elements to be defined independently, but the most common redefinition is for \glstreenamefmt to remove the bold in the name. If the bold is still required for the group heading and navigation elements, then both other commands also need redefining. To simplify matters, all three commands have been defined to use:

\glstreedefaultnamefmt{(text)}

This simply does $\textbf{(text)}$.

This means that if you want to change all three to use a particular style you only need to redefine \glstreedefaultnamefmt, but if you only want to redefine \glstreenamefmt without affecting the other two commands, then you now can.

The separator between groups without headers is given by:

\glstreegroupskip

This defaults to just $\indexspace$ without penalties. This command isn’t used if nogroupskip is set. (The penalties introduced in v1.41 were moved to \glstreegroupheaderskip in v1.42 as they are inappropriate when there’s no header.)

The separator between groups with headers is now given by:

\glstreegroupheaderskip

This defaults to \glstreegroupskip with penalties to deter page breaks after the group heading.

The styles that display the group titles now use:

\glstreePreHeader{(group-label)}{(group-title)}

This does nothing by default and is inserted before the group title. You can redefine it to add the group title to the PDF bookmarks. For example, if the glossary title uses \chapter then:
\renewcommand{\glstreePreHeader}[2]{\%
   \pdfbookmark[1]{#2}{\currentglossary.#1}\%
\}

will insert section-level bookmarks. The use of \currentglossary helps to provide unique bookmark labels in the event of multiple glossaries.

The glossary-tree package provides the commands

\glstreepredesc

and

\glstreechildpredesc

(which both default to a space) and uses them in the tree-like styles, but not for the alt-tree style. The glossaries-extra-stylemods package modifies the alttree style so that it has equivalent hooks:

\glsalttreepredesc

and

\glsalttreechildpredesc

These do nothing by default.

The index-like and tree-like styles insert the pre-location list space with:

\glstreeprelocation \textit{initial:} \glsxtrprelocation

(which defaults to \glsxtrprelocation) for top-level items and

\glstreechildprelocation \textit{initial:} \glstreeprelocation

(which defaults to \glstreeprelocation) for child items.
The styles like treenoname use:

\glstreenonamedesc{⟨entry-label⟩}

to display the pre-description separator, the description and the post-description hook. Similarly for the symbol:

\glstreenamesymbol{⟨entry-label⟩}

The above are just used for top-level entries. Child entries don’t have the name or symbol displayed for the treenoname styles, so there’s only a command for the child description:

\glstreenamechilddesc{⟨entry-label⟩}

For the tree styles (but not the treenoname or alttree styles), the description is displayed using:

\glstreedesc{⟨entry-label⟩}

and the symbol with:

\glstreesymbol{⟨entry-label⟩}

Again the above two commands are just for top-level entries. The child entries use:

\glstreechilddesc{⟨entry-label⟩}

for the description and

\glstreechildsymbol{⟨entry-label⟩}

for the symbol. There are now wrapper commands for \glstreedesc and \glstreechilddesc that check for the description and symbol to determine what separator to use before
8. Defining and Displaying Glossaries

the page list:

\glstreeDescLoc\{⟨entry-label⟩\}{⟨location list⟩}

for top-level entries and

\glstreeChildDescLoc\{⟨entry-label⟩\}{⟨location list⟩}

for sub-entries.

If either the symbol or description is present these will use \glstreeprelocation or \glstreechildprelocation, respectively. Otherwise, both will use:

\glstreeNoDescSymbolPreLocation

The default is a space. This means that you could have, say, a comma followed by a space for terms that are simply an alias, but just have a space for terms that have a description that ends with a full stop (or that just have a symbol without a description) where the comma would be inappropriate.

Version 1.42 has corrected an error that was introduced to v1.41 that caused the name to run into the location list if there was no symbol and no description.

There are some additional commands for use with the alttree style to make it easier to modify. These commands are only defined if the glossary–tree package has already been loaded, which is typically the case unless the notree or nostyles option has been used when loading glossaries.

\gglssetwidest\[⟨level⟩\]{⟨name⟩}

This is like \glssetwidest but performs a global assignment.

\eglssetwidest\[⟨level⟩\]{⟨name⟩}

This is like \glssetwidest but expands ⟨name⟩.

\xglssetwidest\[⟨level⟩\]{⟨name⟩}
This is like `\eglssetwidest` but performs a global assignment.
The following only set the value if \(\langle name\rangle\) is wider than the current value. Local update:

\begin{verbatim}
\glsupdatewidest[\langle level\rangle]{\langle name\rangle}
\end{verbatim}

Global update:

\begin{verbatim}
\gglsupdatewidest[\langle level\rangle]{\langle name\rangle}
\end{verbatim}

Locale update (expands \(\langle name\rangle\)):

\begin{verbatim}
\eglsupdatewidest[\langle level\rangle]{\langle name\rangle}
\end{verbatim}

Global update (expands \(\langle name\rangle\)):

\begin{verbatim}
\xglsupdatewidest[\langle level\rangle]{\langle name\rangle}
\end{verbatim}

The widest entry value can later be retrieved using:

\begin{verbatim}
\glsgetwidestname
\end{verbatim}

which expands to the widest top-level name and:

\begin{verbatim}
\glsgetwidestsubname{\langle level\rangle}
\end{verbatim}

expands to either the widest name for the given hierarchical level or to the widest top-level name, if no widest name set for \(\langle level\rangle\).

Note that if you are using \texttt{bib2gls}, you can use the resource option \texttt{set-widest} which will try to determine the widest name of all the selected entries. This isn’t guaranteed to work as it may depend on fonts or commands that \texttt{bib2gls} can’t replicate, but it should be suitable for names that just consist of text, and can be more efficient than iterating over all the defined entries using \LaTeX.

The command \texttt{\glsfindwidesttoplevelname} provided by \texttt{glossary-tree} has a Camel-Case synonym:

\begin{verbatim}
\glsFindWidestTopLevelName
\end{verbatim}
8. Defining and Displaying Glossaries

Similar commands are also provided. If the optional \textit{⟨glossary labels⟩} is omitted, the list of all non-ignored glossaries is assumed.

\begin{verbatim}
\glsFindWidestUsedTopLevelName[⟨glossary labels⟩]
\end{verbatim}

This has an additional check that the entry has been used. Naturally this is only useful if the glossaries that use the \texttt{alttree} style occur at the end of the document. This command should be placed just before the start of the glossary. (Alternatively, place it at the end of the document and save the value in the auxiliary file for the next run.)

\begin{verbatim}
\glsFindWidestUsedAnyName[⟨glossary labels⟩]
\end{verbatim}

This is like the previous command but if doesn’t check the \texttt{parent} key. This is useful if all hierarchical levels should have the same width for the name.

\begin{verbatim}
\glsFindWidestAnyName[⟨glossary labels⟩]
\end{verbatim}

This is like the previous command but doesn’t check if the entry has been used.

\begin{verbatim}
\glsFindWidestUsedLevelTwo[⟨glossary labels⟩]
\end{verbatim}

This is like \texttt{\glsFindWidestUsedTopLevelName} but also sets the first two sub-levels as well. Any entry that has a great-grandparent is ignored.

\begin{verbatim}
\glsFindWidestLevelTwo[⟨glossary labels⟩]
\end{verbatim}

This is like the previous command but doesn’t check if the entry has been used.

\begin{verbatim}
\glsFindWidestUsedAnyNameSymbol[⟨glossary labels⟩]{⟨register⟩}
\end{verbatim}

This is like \texttt{\glsFindWidestUsedAnyName} but also measures the symbol. The length of the widest symbol is stored in \texttt{⟨register⟩}.

\begin{verbatim}
\glsFindWidestAnyNameSymbol[⟨glossary labels⟩]{⟨register⟩}
\end{verbatim}
8. Defining and Displaying Glossaries

This is like the previous command but it doesn’t check if the entry has been used.

\glsFindWidestUsedAnyNameSymbolLocation[⟨glossary labels⟩]{⟨register1⟩}{⟨register2⟩}

This is like \glsFindWidestUsedAnyNameSymbol but also measures the location list. This requires \glsentrynumberlist. The length of the widest symbol is stored in ⟨register1⟩ and the length of the widest location list is stored in ⟨register2⟩.

\glsFindWidestAnyNameSymbolLocation[⟨glossary labels⟩]{⟨register1⟩}{⟨register2⟩}

This is like the previous command but it doesn’t check if the entry has been used.

\glsFindWidestUsedAnyNameLocation[⟨glossary labels⟩]{⟨register⟩}

This is like \glsFindWidestUsedAnyNameSymbolLocation but doesn’t measure the symbol. The length of the widest location list is stored in ⟨register⟩.

\glsFindWidestAnyNameLocation[⟨glossary labels⟩]{⟨register⟩}

This is like the previous command but doesn’t check if the entry has been used.

The layout of the symbol, description and location list is governed by:

\glstextralttreeSymbolDescLocation{⟨entry-label⟩}{⟨location list⟩}

for top-level entries and

\glstextralttreeSubSymbolDescLocation{⟨entry-label⟩}{⟨location list⟩}

for sub-entries.

There is now a user level command that performs the initialisation for the alttree style:

\glstextralttreeInit

The paragraph indent for subsequent paragraphs in multi-paragraph descriptions is provided
8. Defining and Displaying Glossaries

by the length:

\glsxtrAltTreeIndent

For additional commands that are available with the alttree style, see the documented code (glossaries-extra-code.pdf). See also the accompanying sample files sample-alttree.tex, sample-alttree-sym.tex and sample-alttree-marginpar.tex.

8.7. New Glossary Styles

The glossaries-extra package comes with some new styles. The associated style package needs to be loaded. This can be done with \usepackage but it’s simpler to use the stylemods option. For example:

\usepackage[stylemods=bookindex,style=bookindex]{glossaries-extra}

If you don’t require any of the base styles, use nostyles (but note that some style packages automatically load another style package if it the style builds on an existing one).

8.7.1. glossary-bookindex package

The glossary-bookindex package provides the glossary style bookindex. This is very similar to the mcolindexgroup style but is designed for indexes, so by default only the name and location list are displayed. This style is demonstrated in Example 146 on the following page (using bib2gls). Note that some entries don’t have location lists because they weren’t recorded in the document, but were included as dependencies. See §5.9.1 for dealing with cross-references that may not be required.

The bookindex style only supports a maximum hierarchical level of 2 (top-level, level 1 and level 2). It’s primarily designed for use with bib2gls. It may be used with other indexing options, but some features may not be present and UTF-8 characters may cause a problem with non-Unicode engines in letter group headings or PDF bookmarks. (bib2gls uses numeric identifies by default to avoid these problems, see §8.4.1.)

The number of columns is given by:

\glsxtrbookindexcols initial: 2

which defaults to 2.
### Example 146: The bookindex style

#### Glossary

<table>
<thead>
<tr>
<th>A</th>
<th>lorem, 1, 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>adipiscing, 1</td>
<td>M</td>
</tr>
<tr>
<td>alias-lorem, <em>see</em> lorem</td>
<td>mauris</td>
</tr>
<tr>
<td>amet, 1, <em>see</em> dolor</td>
<td>N</td>
</tr>
<tr>
<td>arcu, 1–3, <em>see also</em> placerat, vitae &amp; curabitur</td>
<td>nulla, 3</td>
</tr>
<tr>
<td>augue, 2</td>
<td>P</td>
</tr>
<tr>
<td>C</td>
<td>nonummy, <em>see also</em> mauris consectetur, 2 vulputate, 1</td>
</tr>
<tr>
<td>curabitur</td>
<td>nulla, 3</td>
</tr>
<tr>
<td>D</td>
<td>P</td>
</tr>
<tr>
<td>dolor</td>
<td>P</td>
</tr>
<tr>
<td>donec, 2</td>
<td>P</td>
</tr>
<tr>
<td>E</td>
<td>pellentesque, <em>see</em> augue placerat, 1</td>
</tr>
<tr>
<td>et, <em>see</em> vulputate</td>
<td>V</td>
</tr>
<tr>
<td>L</td>
<td>vitae</td>
</tr>
<tr>
<td>lectus urna, <em>see</em> nulla</td>
<td></td>
</tr>
</tbody>
</table>

This style uses the `multicols` environment. If the command:

\[\texttt{\textbackslash glsxtbookindexcolspread}\]

isn’t empty then it’s supplied as the optional argument following `\begin{multicols}{⟨n⟩}`. You can switch from `multicols` to `multicols*` by redefining:

\[\texttt{\textbackslash glsxtbookindexmulticolsenv}\]

For example:
8. Defining and Displaying Glossaries

\renewcommand{\glsxtrbookindexmulticolsenv}{multicols*}

Each top-level entry is displayed using:

\glsxtrbookindexname{(entry-label)}

This just does \glossentryname{(entry-label)} by default. For example, if you want the symbol to be included:

\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%
  \ifglshassymbol{#1}{\space (\glossentrysymbol{#1})}%
}%

or if you want the description (if set):

\renewcommand*{\glsxtrbookindexname}[1]{%
  \glossentryname{#1}%
  \ifglshasdesc{#1}{\space \glossentrydesc{#1}\glspostdescription}%
}%

(which picks up the post-description hook).

Alternatively you can use the \glsxtrpostname(category) hook to append information after the name according to the entry’s category.

Sub-entries are displayed using:

\glsxtrbookindexsubname{(entry-label)}

which just defaults to \glsxtrbookindexname{(entry-label)}.

The separator used before the location list for top-level entries is given by:

\glsxtrbookindexprelocation{(entry-label)}

where (entry-label) is the entry’s label. This checks if the location field has been set. If it has, it does:
otherwise it just does \glstxtrprelocation (which defaults to \space) with no comma. If
you’re using bib2gls with \texttt{save-locations=false}, the \texttt{location} field won’t be set.

The separator used before the location list for sub-entries is given by:

\glstxtrbookindexsubprelocation\{⟨entry-label⟩\}

which defaults to \glstxtrbookindexprelocation\{entry-label\}.

The actual location list is encapsulated with:

\glstxtrbookindexlocation\{⟨entry-label⟩\}\{⟨location list⟩\}

for top-level entries and:

\glstxtrbookindexsublocation\{⟨entry-label⟩\}\{⟨location list⟩\}

for sub-entries. These both just do \langle location list \rangle by default.

The separator used between a top-level parent and child entry is given by:

\glstxtrbookindexparentchildsep

This defaults to \texttt{\nopagebreak}.

The separator used between a sub-level parent and child entry is given by:

\glstxtrbookindexparentsubchildsep

This defaults to \glstxtrbookindexparentchildsep.

The separator between top-level entries is given by:

\glstxtrbookindexbetween\{⟨entry1-label⟩\}\{⟨entry2-label⟩\}

This comes after the entry given by \langle entry1-label \rangle, if the entry has no children, or after the
last descendent otherwise, so it always comes immediately before the entry given by \langle entry2-
label \rangle unless the entry occurs at the start of a group. This does nothing by default.
The separator between two level 1 entries is given by:

\glsxtrbookindexsubbetween{⟨entry1-label⟩}{⟨entry2-label⟩}

The separator between two level 2 entries is given by:

\glsxtrbookindexsubsubbetween{⟨entry1-label⟩}{⟨entry2-label⟩}

At the end of each letter group, the following hooks are done in order:

\glsxtrbookindexsubatendgroup{⟨entry-label⟩}

where ⟨entry-label⟩ is the label of the last level 2 entry

\glsxtrbookindexatendgroup{⟨entry-label⟩}

where ⟨entry-label⟩ is the label of the last level 1 entry

\glsxtrbookindexatendgroup{⟨entry-label⟩}

where ⟨entry-label⟩ is the label of the last level 0 entry.

For example, the resource option seealso=omit instructs bib2gls to omit the seealso cross-reference from the location list. (The see cross-reference will still be added unless you also have see=omit.) The seealso cross-reference can instead be appended after the child entries using:

\renewcommand{\glsxtrbookindexatendgroup}[1]{\%
   \glsxtrifhasfield{seealso}{#1}{\glstreesubitem\glsxtruseseealso{#1}}{}}%
\renewcommand{\glsxtrbookindexbetween}[2]{\%
   \glsxtrbookindexatendgroup[#1]}%
\renewcommand{\glsxtrbookindexsubatendgroup}[1]{\%
   \glsxtrifhasfield{seealso}{#1}{\glstreesubsubitem\glsxtruseseealso{#1}}{}}%
This uses \glstreesubitem and \glstreesubsubitem to indent the cross-reference according to the next level down, so the cross-reference for a top-level entry is aligned with the sub-entries, and a level 1 entry has its cross-reference aligned with sub-sub-entries. In the event that a level 2 entry has a cross-reference, this is indented a bit further (but it won’t be aligned with any deeper level as the bookindex style only supports a maximum of two sub-levels).

The bookindex style uses group headings. (If you use bib2gls remember to invoke it with the \--group or \-g switch, see §8.4.1.) The heading will use:

\glsxtrbookindexbookmark{\langle group-title\rangle}{\langle bookmark-name\rangle}

If \pdfbookmark has been defined, this will use that command to bookmark the group title. If section=chapter is set (default if chapters are defined) then this uses level 1 otherwise it uses level 2. You can redefine this command if this isn’t appropriate. If \pdfbookmark hasn’t been defined, this command does nothing.

The group heading is formatted according to:

\glsxtrbookindexformatheader{\langle group-title\rangle}

which is defined as:

\newcommand*{\glsxtrbookindexformatheader}[1]{%  \par{\centering\glstreegroupheaderfmt{#1}\par}%}
where \texttt{\glstreegroupheaderfmt} is provided by the glossary-tree package, which is automatically loaded. Note that the entry names aren't encapsulated with \texttt{\glstreenamefmt}.

The skip after a group header is given by:

\begin{verbatim}
\glsxtrbookindexpregroupskip{⟨skip⟩}
\end{verbatim}

The argument is the skip that would normally be inserted if there wasn't a group header.

The glossary-bookindex package provides some supplementary commands that aren't used by default, but may be used when adjusting the style. These commands should only be used within one of the \texttt{\print...glossary} commands. (That is, they should only be used in glossary styles or in hooks.)

\begin{verbatim}
\glsxtrbookindexmarkentry{⟨entry-label⟩}
\end{verbatim}

This writes information to the aux file that can be read on the next run to obtain the first and last entry on each page of the glossary.

You can display the first entry associated with the current page using:

\begin{verbatim}
\glsxtrbookindexfirstmark{⟨entry-label⟩}
\end{verbatim}

and the last entry associated with the current page using:

\begin{verbatim}
\glsxtrbookindexlastmark{⟨entry-label⟩}
\end{verbatim}

These do nothing if there are no entries marked on the current page (or if the document build isn't up to date).

The entry is formatted using:

\begin{verbatim}
\glsxtrbookindexfirstmarkfmt{⟨entry-label⟩}
\end{verbatim}

for the first instance and

\begin{verbatim}
\glsxtrbookindexlastmarkfmt{⟨entry-label⟩}
\end{verbatim}

for the last.

These commands are designed for use in page headers or footers where the page number is stable. For example, \texttt{\glsxtrbookindexname} can be redefined to mark the current entry:
8. Defining and Displaying Glossaries

If you only want to mark the top-level entries, remember to redefine \glsxtrbookindexname as it defaults to \glsxtrbookindexname:

\renewcommand{\glsxtrbookindexname}[1]{\glsxtrbookindexmarkentry{#1} \glossentryname{#1}}

Then if you’re using fancyhdr you can set the page style to show the first and last entry for the current page with:

\pagestyle{fancy}
\lhead{\thepage}
\lfoot{\glsxtrbookindexfirstmark}
\cfoot{}
\rfoot{\glsxtrbookindexlastmark}

8.7.2. glossary–longextra package

The glossary–longextra package provides additional tabular-like styles similar to those provided by glossary–longbooktabs (which is automatically loaded). These don’t support hierarchical levels except for homographs (level 1 entries with the same name as their parent).

By default, these styles use the longtable environment, but if you know that your glossary won’t span more than a page and you need to use it in a context that’s incompatible with longtable, you can instead setup these styles to use tabular instead. In order to do this you must use:

\GlsLongExtraUseTabulartrue

before the style is set. If you later want to switch back to using longtable for another glossary,
8. Defining and Displaying Glossaries

use:

\GlsLongExtraUseTabularfalse

(or scope \GlsLongExtraUseTabulartrue). Again, the style must be set after this change to the conditional is implemented. You can test this setting with:

\ifGlsLongExtraUseTabular ⟨true⟩\else ⟨false⟩\fi initial: \iffalse

For example:

\GlsLongExtraUseTabulartrue \setglossarystyle{long-name-desc}

or

\GlsLongExtraUseTabulartrue \printunsrtglossary[style=long-name-desc]

If you switch to \texttt{tabular}, the default vertical alignment is obtained from:

\glslongextraTabularVAlign initial: c

This should expand to one of: \texttt{c} (centred), \texttt{t} (top) or \texttt{b} (bottom). The default is \texttt{c}.

For either \texttt{tabular} or \texttt{longtable}, the column titles are formatted according to:

\glslongextraHeaderFmt{⟨text⟩}

which simply does \texttt{\textbf{⟨text⟩}} by default. As with the long-like styles, the header text for the columns are given by the language-sensitive commands: \entryname, \descriptionname, \symbolname and \pagelistname.

Most styles show the name which, as with other predefined styles, also includes the entry item number (if \texttt{entrycounter} is on) and hypertarget anchor. These are all performed for top-level entries with:

\glslongextraNameFmt{⟨entry-label⟩}
This uses \glossentryname, so it supports the post-name hook and associated attributes. Child entries are displayed with:

\glslongextraSubNameFmt{(level)}{(entry-label)}

This includes the sub-entry item number (if subentrycounter is on) and the hypertarget anchor. The actual name isn’t shown by default.

The horizontal alignment for the name column is obtained with:

\glslongextraNameAlign

\initial{l}

This expands to 1 by default.

For styles that show the description, that’s formatted with:

\glslongextraDescFmt{(entry-label)}

for top-level entries, which uses \glossentrydesc and the post-description hook, and

\glslongextraSubDescFmt{(level)}{(entry-label)}

for child entries (which just uses \glslongextraDescFmt).

The horizontal alignment for the description column is obtained with:

\glslongextraDescAlign

This expands to >\{\raggedright\}p{\glsdescwidth} by default. This means ragged-right paragraph style with width given by \glsdescwidth. (See the documentation for the array package for information about this alignment syntax.) If a widest name has been set, \glsdescwidth will be calculated according to the best fit for the given style.

If you are using bib2gls, you may be able to use the set-widest option, otherwise to set the widest name, use:

\glslongextraSetWidest{widest-name}

If you have already used \glssetwidest provided with the alttree style, the default widest name will be obtained from that, but note that only level 0 is supported for the glossary-longextra styles.
8. Defining and Displaying Glossaries

You can update the widest name with:

\glslongextraUpdateWidest\{\langle name\rangle\}

This is like \glslongextraSetWidest but will only set the new value if it’s wider than the current widest name.

Although these styles don’t support hierarchy, the following is provided for child entries:

\glslongextraUpdateWidestChild\{\langle level\rangle\}{\langle name\rangle}

This does nothing by default. If \glslongextraSubNameFmt is redefined to show the child name, then the above command will need to be redefined to use \glslongextraUpdateWidest.

For styles that show the location list, that’s formatted with:

\glslongextraLocationFmt\{\langle entry-label\rangle\}{\langle location list\rangle}

for top-level entries. Child location lists are formatted with:

\glslongextraSubLocationFmt\{\langle level\rangle\}{\langle entry-label\rangle}{\langle location list\rangle}

Both of these simply do the \langle location list\rangle argument.

The horizontal alignment for the location list column is obtained with:

\glslongextraLocationAlign

This expands to >{\raggedright}p{\glspagelistwidth} by default. This means ragged-right paragraph style with width given by \glspagelistwidth.

For styles that show the symbol (in addition to the name), that’s formatted with:

\glslongextraSymbolFmt\{\langle entry-label\rangle\}

for top-level entries. This simply uses \glossentrysymbol. Child entries use:

\glslongextraSubSymbolFmt\{\langle level\rangle\}{\langle entry-label\rangle}
which uses \glslongextraSymbolFmt.

The horizontal alignment for the symbol column (except for the long-sym-desc and long-desc-sym styles) is obtained with:

\glslongextraSymbolAlign initial: c

This expands to c by default.

Top-level group headings are formatted with:

\glslongextraGroupHeading{(number columns)}{(group-label)}

The first argument is the total number of columns in the table. For example, 2 for the long-name-desc style or 3 for the long-name-sym-desc style. The second argument is the group’s label (not the title). This command does nothing by default. (If you are using bib2gls, remember that you need to use the --group or -g switch to support groups.)

Sub-level groups are only supported with the “unsrt” family of commands (see §8.4.1). When they are supported, the heading will be formatted with:

\glslongextraSubGroupHeading{(number columns)}{(prev group level)}{(group level)}{(parent-entry-label)}{(group-label)}

The styles are sub-divided below into the set of elements that are shown in each column, which may consist of: name, symbol, description or location list. There will be blank cells if any of the corresponding fields have not been set or if the location list has been suppressed.

### 8.7.2.1. Name and Description Only

These styles don’t display the symbol or location list, regardless of whether or not they have been set. In each case, the style starts with:

\glslongextraSetDescWidth

which updates \glsdescwidth according to the widest name, identified with \glslongextraSetWidest. The column header text is also taken into account. If a widest name hasn’t been set and the column header is shorter than one or more names, the description column may be too wide. The value of \glsdescwidth is calculated as \linewidth - 4\tabcolsep - W, where W is the width of the widest name.

If you want to set \glsdescwidth to a specific value, then redefine \glslongextraSet-
DescWidth with the desired length assignment.

| long-name-desc |

This has two columns: the name on the left and the description on the right. The table header is given by:

\texttt{\textbackslash \textit{\textbackslash longextraNameDescTabularHeader}}

which shows the column headers with horizontal rules. The table footer is given by:

\texttt{\textbackslash \textit{\textbackslash longextraNameDescTabularFooter}}

which just does a horizontal rule. With \texttt{longtable}, the table header and footer are set with:

\texttt{\textbackslash \textit{\textbackslash longextraNameDescHeader}}

which uses the above header and footer commands.

| long-desc-name |

This has two columns: the name on the right and the description on the left. The table header is given by:

\texttt{\textbackslash \textit{\textbackslash longextraDescNameTabularHeader}}

which shows the column headers with horizontal rules. The table footer is given by:

\texttt{\textbackslash \textit{\textbackslash longextraDescNameTabularFooter}}

which just does a horizontal rule. With \texttt{longtable}, the table header and footer are set with:

\texttt{\textbackslash \textit{\textbackslash longextraDescNameHeader}}

which uses the above header and footer commands.
8.7.2.2. Name, Symbol and Description Only

These styles don’t show the location list. In each case, the style starts with:

\glslongextraSymSetDescWidth

which updates \glsdescwidth according to the widest name, identified with \glslongextraSetWidest. This starts by calculating \glsdescwidth with \glslongextraSetDescWidth and then subtracts the width of the symbol column header text (which is assumed to be the widest text in that column).

If you want to set \glsdescwidth to a specific value, then redefine \glslongextraSymSetDescWidth with the desired length assignment.

\begin{itemize}
\item \texttt{long-name-desc-sym}
\end{itemize}

This has three columns: the name on the left, the description in the middle and the symbol on the right. The table header is given by:

\glslongextraNameDescSymTabularHeader

which shows the column headers with horizontal rules. The table footer is given by:

\glslongextraNameDescSymTabularFooter

which just does a horizontal rule. With longtable, the table header and footer are set with:

\begin{itemize}
\item \texttt{long-name-sym-desc}
\end{itemize}

This has three columns: the name on the left, the symbol in the middle and the description on the right. The table header is given by:

\glslongextraNameSymDescTabularHeader
8. Defining and Displaying Glossaries

which shows the column headers with horizontal rules. The table footer is given by:

\glslongextraNameSymDescTabularFooter

which just does a horizontal rule. With longtable, the table header and footer are set with:

\glslongextraNameSymDescHeader

which uses the above header and footer commands.

long-sym-desc-name

This has three columns: the name on the right, the description in the middle and the symbol on the left. The table header is given by:

\glslongextraSymDescNameTabularHeader

which shows the column headers with horizontal rules. The table footer is given by:

\glslongextraSymDescNameTabularFooter

which just does a horizontal rule. With longtable, the table header and footer are set with:

\glslongextraSymDescNameHeader

which uses the above header and footer commands.

long-desc-sym-name

This has three columns: the name on the right, the symbol in the middle and the description on the left. The table header is given by:

\glslongextraDescSymNameTabularHeader
which shows the column headers with horizontal rules. The table footer is given by:

\glslongextraDescSymNameTabularFooter

which just does a horizontal rule. With longtable, the table header and footer are set with:

\glslongextraDescSymNameHeader

which uses the above header and footer commands.

8.7.2.3. Name, Description and Location Only

These styles don’t display the symbol, regardless of whether or not the symbol field has been set. In each case, the style starts with:

\glslongextraLocSetDescWidth

which updates \glsdescwidth according to the widest name, identified with \glslongextraSetWidest. This starts by calculating \glsdescwidth with \glslongextraSetDescWidth and then subtracts $2\tabcolsep - \glspagelistwidth$.

If you want to set \glsdescwidth to a specific value, then redefine \glslongextraLocSetDescWidth with the desired length assignment.

long-name-desc-loc

This has three columns: the name on the left, the description in the middle and the location list on the right. The table header is given by:

\glslongextraNameDescLocationTabularHeader

which shows the column headers with horizontal rules. The table footer is given by:

\glslongextraNameDescLocationTabularFooter

which just does a horizontal rule. With longtable, the table header and footer are set with:

\glslongextraNameDescLocationHeader
which uses the above header and footer commands.

\begin{longtable}{llll}
\caption{Location, Description, Name}
\end{longtable}

This has three columns: the name on the right, the description in the middle and the location list on the left. The table header is given by:

\begin{verbatim}
\glslongextralocationdescname TabularHeader
\end{verbatim}

which shows the column headers with horizontal rules. The table footer is given by:

\begin{verbatim}
\glslongextralocationdescname TabularFooter
\end{verbatim}

which just does a horizontal rule. With \texttt{longtable}, the table header and footer are set with:

\begin{verbatim}
\glslongextralocationdescname Header
\end{verbatim}

which uses the above header and footer commands.

\subsection*{8.7.2.4. Name, Description, Symbol and Location}

These styles show the name, description, symbol and location list. In each case, the style starts with:

\begin{verbatim}
\glslongextrasymlocsetdescwidth
\end{verbatim}

which updates \texttt{\glsdescwidth} according to the widest name, identified with \texttt{\glslongextrasetwidest}. This starts by calculating \texttt{\glsdescwidth} with \texttt{\glslongextrasymsymsetdescwidth} and then subtracts $2\texttt{\tabcolsep} - \texttt{\glspagelistwidth}$.

If you want to set \texttt{\glsdescwidth} to a specific value, then redefine \texttt{\glslongextrasymlocsetdescwidth} with the desired length assignment.

\begin{longtable}{llll}
\caption{Name, Description, Symbol and Location}
\end{longtable}

This has four columns, from left to right: the name, description, symbol and the location list.
The table header is given by:

```
\longextraNameDescSymLocationTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\longextraNameDescSymLocationTabularFooter
```

which just does a horizontal rule. With `longtable`, the table header and footer are set with:

```
\longextraNameDescSymLocationHeader
```

which uses the above header and footer commands.

```
long-name-sym-desc-loc
```

This has four columns, from left to right: the name, symbol, description and the location list. The table header is given by:

```
\longextraNameSymDescLocationTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\longextraNameSymDescLocationTabularFooter
```

which just does a horizontal rule. With `longtable`, the table header and footer are set with:

```
\longextraNameSymDescLocationHeader
```

which uses the above header and footer commands.

```
long-loc-sym-desc-name
```

This has four columns, from left to right: the location list, symbol, description and the name.
8. Defining and Displaying Glossaries

The table header is given by:

```
\glslongextraLocationSymDescNameTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\glslongextraLocationSymDescNameTabularFooter
```

which just does a horizontal rule. With longtable, the table header and footer are set with:

```
\glslongextraLocationSymDescNameHeader
```

which uses the above header and footer commands.

```
\longloc\text{-}\text{desc}\text{-}\text{sym}\text{-}\text{name}
```

This has four columns, from left to right: the location list, description, symbol and the name. The table header is given by:

```
\glslongextraLocationDescSymNameTabularHeader
```

which shows the column headers with horizontal rules. The table footer is given by:

```
\glslongextraLocationDescSymNameTabularFooter
```

which just does a horizontal rule. With longtable, the table header and footer are set with:

```
\glslongextraLocationDescSymNameHeader
```

which uses the above header and footer commands.

### 8.7.2.5. Symbol and Description Only

These are two-column styles designed to show only the symbol and description. However, if the symbol isn’t set then the name will be used instead. If this occurs, you may need to change the width of the description column.
The horizontal alignment for the symbol column is obtained with:

\glslongextraSymbolNameAlign

which expands to l by default. Note that this is different from the alignment used for styles like long-name-sym-desc.

These styles have the entry item number (if entrycounter is on) and the hypertarget anchor (if enabled) in the symbol column since there’s no name shown (unless the symbol is missing). These are all performed by for top-level entries by:

\glslongextraSymbolTargetFmt{⟨entry-label⟩}

The symbol is formatted according to \glslongextraSymbolFmt. Child entries use:

\glslongextraSubSymbolTargetFmt{⟨level⟩}{⟨entry-label⟩}

Unlike \glslongextraSubNameFmt this shows the field value (formatted with \glslongextraSymbolFmt).

The following commands use the above if the symbol field is set, otherwise they show the name.

\glslongextraSymbolOrName{⟨entry-label⟩}

Shows the symbol, if set, or the name otherwise, with the target. Child entries use:

\glslongextraSubSymbolOrName{⟨level⟩}{⟨entry-label⟩}

Shows the symbol with \glslongextraSubSymbolTargetFmt, if set, or the name otherwise, with the target.

In each case, the style starts with:

\glslongextraSymNoNameSetDescWidth

which calculates \glsdescwidth as \linewidth – 4 \tabcolsep – W, where W is the width of the symbol column header. Note that this assumes the content of the symbol column isn’t wider than the column header.

If you want to set \glsdescwidth to a specific value, then redefine \glslongextraSymNoNameSetDescWidth with the desired length assignment. For example, if you have a
8. Defining and Displaying Glossaries

mixture of entries with symbols and some without, which means that there will be a name shown that’s wider than the symbol column header, then set the widest name (for example, with the `set-widest` resource option) and add the following redefinition:

\renewcommand{\glslongextrasymsymnomenamedescwidth}{% \glslongextrasetdescwidth}

Note that, in this case, if you don’t set the widest name then the description column will end up even wider (and therefore cause the table to be even wider) if the name header is narrower than the symbol header.

```
long-sym-desc
```

The symbol is in the left column (or the name, if the symbol isn’t set). The description is in the right. The location list isn’t shown. The table header is given by:

\glslongextrasymsymdesctabularheader

which shows the column headers with horizontal rules. The table footer is given by:

\glslongextrasymsymdesctabularfooter

which just does a horizontal rule. With `longtable`, the table header and footer are set with:

\glslongextrasymsymdescheader

which uses the above header and footer commands.

```
long-desc-sym
```

The symbol is in the right column (or the name, if the symbol isn’t set). The description is in the left. The location list isn’t shown. The table header is given by:

\glslongextradescsymtabularheader
which shows the column headers with horizontal rules. The table footer is given by:

\[\text{\texttt{\textbackslash glslongextraDescSymTabularFooter}}\]

which just does a horizontal rule. With longtable, the table header and footer are set with:

\[\text{\texttt{\textbackslash glslongextraDescSymHeader}}\]

which uses the above header and footer commands.

**8.7.2.6. Abbreviations Only**

These styles are designed for abbreviations. They display the short and long forms, rather than the name and description, although these may happen to match. They are primarily intended for mini-glossaries or similar summary lists.

Although these styles don’t show the name or description, they still use some of the name and description settings provided by glossary-longextra. The column for the short form uses the same alignment as for the name columns (\glslongextraNameAlign). The column for the long form uses the same alignment as for the description columns (\glslongextraDescAlign) and has the width set to \glsdescwidth. However, the name and description formatting commands or attributes (such as \glsnamefont, \glossnamefont or \glossname) aren’t used as the formatting is left to the abbreviation style.

If the short field hasn’t been set, the short column will show the name instead, and if the long field hasn’t been set, the long column will show the description instead (using the same commands as for styles like long-name-desc, which do use the associated formatting commands and attributes).

These styles use the following commands:

\[\text{\texttt{\textbackslash glslongextraShortHeader}}\]

*initial: \entryname*

The header for the column showing the short form. This is defined as:

\[\text{\texttt{\newcommand{\glslongextraShortHeader}{\entryname}}}\]

\[\text{\texttt{\glslongextraLongHeader}}\]

*initial: \descriptionname*

The header for the column showing the long form. This is defined as:

\[\text{\texttt{\newcommand{\glslongextraLongHeader}{\descriptionname}}}\]
Defining and Displaying Glossaries

\glslongextraShortTargetFmt{⟨entry-label⟩}

This governs the way that the short form should be displayed, including the target. This is defined as:

\newcommand{\glslongextraShortTargetFmt}[1]{
\glsentryitem{#1}\glstarget{#1}{\glsxtrshort[noindex,hyper=false]{#1}}\glsxtrpostnamehook{#1}
}

Note that the post-name hook is included.

\glslongextraLongFmt{⟨entry-label⟩}

This governs the way that the long form should be displayed. This is defined as:

\newcommand{\glslongextraLongFmt}[1]{\glsxtrlong[noindex,hyper=false]{#1}\glspostdescription}

Note that the post-description hook is included.

\glslongextraSubShortTargetFmt{⟨level⟩}{⟨entry-label⟩}

This governs the way that the short form for child entries should be displayed, including the target. This is defined as:

\newcommand{\glslongextraSubShortTargetFmt}[2]{\glssubentryitem{#2}\glstarget{#2}{\glsxtrshort[noindex,hyper=false]{#2}}\glsxtrpostnamehook{#2}}

\glslongextraSubLongFmt{⟨level⟩}{⟨entry-label⟩}

This governs the way that the long form for child entries should be displayed. This is defined as:

\newcommand{\glslongextraSubLongFmt}[2]{\glslongextraLongFmt{#2}}
This is used to compute the value of \texttt{\glsdescwidth} and assumes that none of the short forms are wider than \texttt{\glslongextraShortHeader}.

\textit{abbr-short-long}

A two column style. The short form is in the left column. The long form is in the right. The location list isn’t shown.

The table header is given by:

\texttt{\glslongextraShortLongTabularHeader}

which shows the column headers with horizontal rules.

The table footer is given by:

\texttt{\glslongextraShortLongTabularFooter}

which just does a horizontal rule. With \texttt{longtable}, the table header and footer are set with:

\texttt{\glslongextraShortLongHeader}

which uses the above header and footer commands.

\textit{abbr-long-short}

A two column style. The short form is in the right column. The long form is in the left. The location list isn’t shown.

The table header is given by:

\texttt{\glslongextraLongShortTabularHeader}

which shows the column headers with horizontal rules.

The table footer is given by:

\texttt{\glslongextraLongShortTabularFooter}
8. Defining and Displaying Glossaries

which just does a horizontal rule. With longtable, the table header and footer are set with:

\glslongextraLongShortHeader

which uses the above header and footer commands.

### 8.7.2.7. Custom Fields

These styles allow one, two or three custom columns in addition to the name column. The “custom1” styles indicate one custom column, the “custom2” styles indicate two custom columns, and the “custom3” styles indicate three custom columns. Some styles also include the description column. These styles don’t display the location. However, if you are using bib2gls you can set one of the custom fields to location, but if you have long location lists you may need to change the corresponding alignment command to switch to a paragraph column.

The “first custom column” means the first of the custom columns, which may not be the first column in the table. Similarly the “second custom column” means the second of the custom columns (if supported by the style), and the “third custom column” means the third of the custom columns (if supported by the style).

\glslongextraCustomIField \textit{initial: useri}

Expands to the internal field label for the first custom column. This will be used in the “custom1”, “custom2” and “custom3” styles.

\glslongextraCustomIIField \textit{initial: userii}

Expands to the internal field label for the second custom column. This will be used in the “custom2” and “custom3” styles.

\glslongextraCustomIIIField \textit{initial: useriii}

Expands to the internal field label for the third custom column. This will be used in the “custom3” style.
Expands to the header text for the first custom column. The default definition is:

\MFUsentencecase{\glslongextraCustomIField}

which means that it will be “Useri” by default, which is unlikely to be appropriate, but it may be suitable if you change the first custom field.

\glslongextraCustomIIHeader

Expands to the header text for the second custom column. The default likewise simply applies sentence case to the internal field label.

\glslongextraCustomIIIHeader

Expands to the header text for the third custom column. The default likewise simply applies sentence case to the internal field label.

\glslongextraCustomIFmt{⟨entry-label⟩}

This is used to format each top-level entry in the first custom column. The default definition is:

\glsxtrusefield{⟨entry-label⟩}{\glslongextraCustomIField}

\glslongextraSubCustomIFmt{⟨level⟩}{⟨entry-label⟩}

This is used to format each sub-entry in the first custom column. The default definition is:

\glslongextraCustomIFmt{⟨entry-label⟩}

\glslongextraCustomIIFmt{⟨entry-label⟩}

This is used to format each top-level entry in the second custom column. The default definition is:

\glsxtrusefield{⟨entry-label⟩}{\glslongextraCustomIIField}
This is used to format each sub-entry in the second custom column. The default definition is:

\begin{verbatim}
\textbf{\textglslongextraSubCustomIIFmt{\langle level\rangle}{\langle entry-label\rangle}}
\end{verbatim}

This is used to format each top-level entry in the third custom column. The default definition is:

\begin{verbatim}
\textbf{\textglslongextraCustomIIIFmt{\langle entry-label\rangle}}
\end{verbatim}

This is used to format each sub-entry in the third custom column. The default definition is:

\begin{verbatim}
\textbf{\textglslongextraSubCustomIIIFmt{\langle level\rangle}{\langle entry-label\rangle}}
\end{verbatim}

Expands to the alignment specifier for the first custom column.

\begin{verbatim}
\textbf{\textglslongextraCustomIAlign }
\end{verbatim}

\begin{verbatim}
\textbf{initial: l}
\end{verbatim}

Expands to the alignment specifier for the second custom column.

\begin{verbatim}
\textbf{\textglslongextraCustomIIAlign }
\end{verbatim}

\begin{verbatim}
\textbf{initial: l}
\end{verbatim}

Expands to the alignment specifier for the third custom column.

\begin{verbatim}
\textbf{\textglslongextraCustomIIIAlign }
\end{verbatim}

\begin{verbatim}
\textbf{initial: l}
\end{verbatim}

\begin{verbatim}
\textbf{\textglslongextraCustomTabularFooter}
\end{verbatim}
The footer used for all the custom styles. The default definition simply does \bottomrule.

\glslongextraNameCustomITabularHeader

The table header for the long-name-custom1 style (which has two columns). This command and the previous command are used in the following.

\glslongextraNameCustomIHeader

The longtable header and footer markup for the long-name-custom1 style.

\glslongextraCustomINameTabularHeader

The table header for the long-custom1-name style (which has two columns). Used in the following.

\glslongextraCustomINameHeader

The longtable header and footer markup for the long-custom1-name style.

\glslongextraNameCustomIIITabularHeader

The table header for the long-name-custom2 style (which has three columns). Used in the following.

\glslongextraNameCustomIIHeader

The longtable header and footer markup for the long-name-custom2 style.

\glslongextraCustomIINameTabularHeader

The table header for the long-custom2-name style (which has three columns). Used in the following.

\glslongextraCustomIINameHeader
8. Defining and Displaying Glossaries

The `longtable` header and footer markup for the `long-name-custom2-name` style.

\texttt{\textbackslash glslongextraNameCustomIIITabularHeader}

The table header for the `long-name-custom3` style (which has four columns). Used in the following.

\texttt{\textbackslash glslongextraNameCustomIIIHeader}

The `longtable` header and footer markup for the `long-name-custom3` style.

\texttt{\textbackslash glslongextraCustomIIINameTabularHeader}

The table header for the `long-custom3-name` style (which has four columns). Used in the following.

\texttt{\textbackslash glslongextraCustomIIINameHeader}

The `longtable` header and footer markup for the `long-custom3-name` style.

long-name-custom1

A two column style with the name in the first column and the first custom field in the second. For top-level entries, the name is formatted with \texttt{\textbackslash glslongextraNameFmt} and the custom field is formatted with \texttt{\textbackslash glslongextraCustomIFmt}. Sub-entries use \texttt{\textbackslash glslongextraSubNameFmt} and \texttt{\textbackslash glslongextraSubCustomIFmt}.

long-custom1-name

As `long-name-custom1` but with the name in the last column.

long-name-custom2

A three column style with the name in the first column, the first custom field in the second and the second custom field in the third. As `long-name-custom1`, but additionally the second custom field is formatted with \texttt{\textbackslash glslongextraCustomIIFmt} for top-level entries and with
8. Defining and Displaying Glossaries

```
\texttt{glslongextraSubCustomIIFmt} for child-entries.
```

As `long-name-custom2` but with the name in the last column.

```
\texttt{long-name-custom3}
```

A four column style with the name in the first column, the first custom field in the second, the second custom field in the third, and the third custom field in the fourth. As `long-name-custom2`, but additionally the third custom field is formatted with `\texttt{glslongextraCustomIIIFmt}` for top-level entries and with `\texttt{glslongextraSubCustomIIFmt}` for child-entries.

```
\texttt{long-name-custom3-name}
```

As `long-name-custom3` but with the name in the last column.

The following styles also have a description column, which uses `\texttt{glslongextraDescAlign}` for the column alignment. These styles attempt to calculate an appropriate width for `\texttt{glsdescwidth}`.

```
\texttt{glslongextraCustomISetDescWidth}
```

Sets `\texttt{glsdescwidth}` for the `long-name-custom1-desc` style. This first uses `\texttt{glslongextraSetDescWidth}` to calculate the width $W$ if there were only a name and description column. It then measures the width of the first custom column header $w$ and sets `\texttt{glsdescwidth}` to $w - 2\texttt{tabcolsep} - w$. This assumes that the first custom column header is wider than the value of each entry’s first custom field. If this isn’t the case, then you will need to redefine this command as appropriate.

```
\texttt{glslongextraCustomIISetDescWidth}
```

Sets `\texttt{glsdescwidth}` for the `long-name-custom2-desc` style. This first uses `\texttt{glslongextraCustomIISetDescWidth}` to calculate the width $W$ if there were only a name column, first custom column, and description column. It then measures the width of the second custom column header $w$ and sets `\texttt{glsdescwidth}` to $w - 2\texttt{tabcolsep} - w$. This assumes that the second custom column header is wider than the value of each entry’s second custom field. If
this isn’t the case, then you will need to redefined this command as appropriate.

```latex
\glslongextraCustomIIISetDescWidth
```

Sets \glssdescwidth for the long-name-custom3-desc style. This first uses \glslongextraCustomIIISetDescWidth to calculate the width \( W \) if there were only a name column, first custom column, second custom column, and description column. It then measures the width of the third custom column header \( w \) and sets \glssdescwidth to \( w - 2\tabcolsep - w \). This assumes that the third custom column header is wider than the value of each entry’s third custom field. If this isn’t the case, then you will need to redefined this command as appropriate.

```latex
\glslongextraNameCustomIDescTabularHeader
```

The table header for the long-name-custom1-desc style (which has three columns). Used in the following.

```latex
\glslongextraNameCustomIDescHeader
```

The longtable header and footer markup for the long-name-custom1-desc style.

```latex
\glslongextraDescCustomINameTabularHeader
```

The table header for the long-desc-custom1-name style (which has three columns). Used in the following.

```latex
\glslongextraDescCustomINameHeader
```

The longtable header and footer markup for the long-desc-custom1-name style.

```latex
\glslongextraNameCustomIIDescTabularHeader
```

The table header for the long-name-custom2-desc style (which has four columns). Used in the following.

```latex
\glslongextraNameCustomIIDescHeader
```
8. Defining and Displaying Glossaries

The longtable header and footer markup for the long-name-custom2-desc style.

\texttt{\textbackslash glslongextraDescCustomIINameTabularHeader}

The table header for the long-desc-custom2-name style (which has four columns). Used in the following.

\texttt{\textbackslash glslongextraDescCustomIINameHeader}

The longtable header and footer markup for the long-desc-custom2-name style.

\texttt{\textbackslash glslongextraNameCustomIIIDescTabularHeader}

The table header for the long-name-custom3-desc style (which has five columns). Used in the following.

\texttt{\textbackslash glslongextraNameCustomIIIDescHeader}

The longtable header and footer markup for the long-name-custom3-desc style.

\texttt{\textbackslash glslongextraDescCustomIIINameTabularHeader}

The table header for the long-desc-custom3-name style (which has five columns). Used in the following.

\texttt{\textbackslash glslongextraDescCustomIIINameHeader}

The longtable header and footer markup for the long-desc-custom3-name style.

\texttt{long-name-custom1-desc}

A three column style with the name in the first column, the first custom field in the second, and the description in the third. This is like long-name-custom1 but additionally has the description column formatted as per long-name-desc.

\texttt{long-desc-custom1-name}
As long-name-custom1-desc but the name and description columns are swapped.

- **long-name-custom2-desc**

A four column style with the name in the first column, the first custom field in the second, the second custom field in the third, and the description in the fourth. This is like long-name-custom2 but additionally has the description column formatted as per long-name-desc.

- **long-desc-custom2-name**

As long-name-custom2-desc but the name and description columns are swapped.

- **long-name-custom3-desc**

A five column style with the name in the first column, the first custom field in the second, the second custom field in the third, the third custom field in the fourth, and the description in the fifth. This is like long-name-custom3 but additionally has the description column formatted as per long-name-desc.

- **long-desc-custom3-name**

As long-name-custom3-desc but the name and description columns are swapped.

### 8.7.3. glossary-topic package

The glossary-topic package provides glossary styles designed for hierarchical glossaries where the top-level entries are topic titles. This package automatically loads the multicol package. If the glossary-tree package is also loaded then commands like `{\glssetwidest` can be used on these styles in much the same way as for the alttree style. If a widest value isn’t set then these styles behave more like the tree style.

This package provides styles designed for glossaries that are lists of topics. That is, the top-level entries are considered topic titles (which may or may not have an associated symbol or description) and the sub-entries are items within that topic. By default the location list isn’t shown for the top-level entries but is shown after the description for sub-entries (unless suppressed with `nonumberlist` or `save-locations=false`).

The following styles are provided:

- **topic**


This style is similar to the tree style but the indentation doesn’t start until the second sub-item level. The top-level entries have the name displayed in a larger font with the description following in a new paragraph (see Example 147). This style doesn’t support the nogroupskip setting.

Example 147: The topic style

Glossary

Pictograph
Picture or symbol representing a word or phrase.

- copy.
- cut.
- edit.
- paste.

Symbols
Mathematical constants or functions.

- constant  a fixed quantity or numerical value.
  - \( \pi \) ratio of the length of the circumference of a circle to its diameter.
  - \( \sqrt{2} \) Pythagoras’ constant.
- function  a rule that assigns a value to every element of the domain.
  - \( \cos(x) \) cosine.
  - \( \ln(x) \) natural logarithm.
- parameter  a constant or variable that distinguishes a specific form.
  - \( x \) the abscissa value.
  - \( y \) the ordinate value.
  - \( z \) the applicate value.

This style is like the topic style but the sub-entries are placed inside a multicols environment (unlike styles such as mcoltree where the entire glossary content is within a single multicols environment). The environment name is supplied in the value of the command:

\glstopicColsEnv

This defaults to multicols. You can change this to the starred form. For example:
8. Defining and Displaying Glossaries

\renewcommand{\glstopicColsEnv}{multicols*}

The number of columns is given by the command:

\glstopicCols

This expands to 2, by default. This style is demonstrated in Example 148.

Example 148: The topicmcols style

**Glossary**

**Pictograph**
Picture or symbol representing a word or phrase.

- copy.
- cut.
- edit.
- paste.

**Symbols**
Mathematical constants or functions.

constant a fixed quantity or numerical value.

- $\pi$ ratio of the length of the circumference of a circle to its diameter.
- $\sqrt{2}$ Pythagoras’ constant.

function a rule that assigns a value to every element of the domain.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\cos(x)$</td>
<td>cosine.</td>
</tr>
<tr>
<td>$\ln(x)$</td>
<td>natural logarithm.</td>
</tr>
</tbody>
</table>

Both styles can have a widest name set like the alttree style, using the commands provided by glossary-tree and glossaries-extra-stylemods or with the `set-widest` resource option. If a widest name is set, then the sub-entry names will be placed in a box of the given width otherwise they won’t be placed in a box. In Example 149 on the following page, the widest names have been set for level 1 and level 2 using:
8. Defining and Displaying Glossaries

\glssetwidest[1]{parameter}
\glssetwidest[2]{\glsentryname{cosine}}

Note that this doesn’t change the indentation at the start of the level 2 items to match the width of the level 1 widest name.

Example 149: The \texttt{topicmcols} style with the widest name set

\section*{Glossary}

\subsection*{Pictograph}
Picture or symbol representing a word or phrase.

\begin{itemize}
  \item \texttt{copy}.
  \item \texttt{cut}.
  \item \texttt{edit}.
  \item \texttt{paste}.
\end{itemize}

\subsection*{Symbols}
Mathematical constants or functions.

\begin{verbatim}
constant a fixed quantity or numerical value.
\pi ratio of the length of the circumference of a circle to its diameter.
\sqrt{2} Pythagoras’ constant.
function a rule that assigns a value to every element of the domain.
\end{verbatim}

\begin{verbatim}
\cos(x) cosine.
\ln(x) natural logarithm.
\end{verbatim}

\begin{verbatim}
\texttt{parameter} a constant or variable that distinguishes a specific form.
x the abscissa value.
y the ordinate value.
z the applicate value.
\end{verbatim}

Both of the above styles use the following commands.

\glistopicParIndent

This command is a length that’s used for the paragraph indentation in any multi-paragraph description for top-level entries, but not for the first paragraph (at the start of the description)
8. Defining and Displaying Glossaries

which isn’t indented.

\glstopicSubIndent

This command is a length that’s used to calculate the hanging indentation for sub-entries. The level 1 sub-entries don’t indent the name. Level \( n \) sub-entries have the name indented by \((n-1) \times \text{\glstopicSubIndent}\). The hanging indent depends on whether or not a widest name has been set for the level.

There is also a length for additional indentation used in the second paragraph onwards for child entries with multi-paragraph descriptions:

\glstopicSubItemParIndent

This is initialised to \texttt{\parindent} when \texttt{glossary-topic} is loaded.

\glstopicInit

This hook is used at the start of the glossary. It does nothing by default.

Although the styles don’t support letter groups by default, if you have many topics (top-level entries) and you feel that it would help the reader to divide them up into headed letter groups, you can redefine:

\glstopicGroupHeading\{⟨\texttt{group-label}⟩\}

This does nothing by default. If you want to redefine it, you can fetch the title corresponding to the group label with \texttt{\glsxtrgetgrouptitle}. For example:

\renewcommand*{\glstopicGroupHeading}[1]{% 
\glsxtrgetgrouptitle{#1}{\thisgrptitle}% 
\section*{\thisgrptitle}%
} 

Remember that if you are using \texttt{bib2gls}, you will need the \texttt{--group} or \texttt{-g} switch to support groups (see §8.4.1).

Sub-groups are only available with \texttt{bib2gls} and the \texttt{group-level} option. If they are
8. Defining and Displaying Glossaries

supported, sub-group headings are formatted according to:

\glstopicSubGroupHeading\{\langle prev group level\rangle\}\{\langle group level\rangle\}\{\langle parent entry\rangle\}\{\langle group-label\rangle\}

This formats the sub-group heading. Note that unlike \glstopicGroupHeading this command does actually format the sub-group heading by default. This means that if you use group-level=all, the top-level groups won’t be displayed, but the sub-groups will be.

Top-level entries are formatted according to:

\glstopicItem\{\langle entry-label\rangle\}\{\langle location list\rangle\}

This formats the name and (if provided) the symbol. The description (if set) will follow in a new paragraph. At the start of \glstopicItem, a vertical space is added with:

\glstopicPreSkip

which defaults to \medskip. There is then a hook:

\glstopicMarker\{\langle entry-label\rangle\}

which does nothing by default, but may be redefined. For example, to add a line to the table of contents. The name and symbol are set in the form of a title using:

\glstopicTitle\{\langle entry-label\rangle\}

This uses \Glossentryname which uses sentence case. If there’s a symbol, this is added in parentheses. Both name and symbol (if present) are encapsulated by:

\glstopicTitleFont\{\langle text\rangle\}

This uses a bold, large font by default.

If the entry has the description key set (tested with \ifglshasdesc) then a paragraph break is inserted followed by:

\glstopicMidSkip

480
which defaults to \smallskip. This is followed by the description which is formatted according to:

\glstopicDesc{⟨entry-label⟩}

This just does \Glossentrydesc{⟨entry-label⟩} followed by the post-description hook.

\glstopicPostSkip

regardless of whether or not the description was displayed. This defaults to \smallskip.

This is then followed by:

\glstopicLoc{⟨entry-label⟩}{⟨location list⟩}

which may be used to display the location list, but does nothing by default.

The sub-entries first set up the paragraph and hanging indentations using:

\glstopicAssignSubIndent{⟨level⟩}

This uses:

\glstopicAssignWidest{⟨level⟩}

to determine if a widest name has been set for the given level.

The sub-entry has its information displayed using:

\glstopicSubItem{⟨level⟩}{⟨entry-label⟩}{⟨location list⟩}

This encapsulates the name with:

\glstopicSubNameFont{⟨text⟩}
8. Defining and Displaying Glossaries

By default this just uses `\textbf{. This is followed by:

```
\textbf{\glstopicSubItemSep}
```

which defaults to `\quad`. The name and separator are passed in the \textit{text} argument of:

```
\textbf{\glstopicSubItemBox{⟨level⟩}{⟨text⟩}}
```

If a widest name was set for the given level, this will put \textit{text} inside a box of that width otherwise it just does \textit{text}.

This is followed by the symbol in parentheses if set. Then, if the description is set, the description and post-description hook are displayed followed by:

```
\textbf{\glstopicSubPreLocSep}
```

(This command isn’t used if the description isn’t set.)

Finally the location list is displayed using:

```
\textbf{\glstopicSubLoc{⟨entry-label⟩}{⟨location list⟩}}
```

which just does \textit{location} by default.

8.7.4. glossary-table package

The glossary-table package is new to version 1.49. It automatically loads the longtable, array and booktabs packages. If you want to use \texttt{\glspenaltygroupskip} for the group skip, you will need to also load glossary-longbooktabs.

This package is designed specifically for use with bib2gls. It can be used to create a supplemental glossary with other indexing options, but the entries will be listed in order of definition and no child entries will be shown.

The glossary-table package doesn’t provide any general purpose styles, but instead provides one highly customized style (table), which is designed to work with a supplied command:

```
\texttt{\printunsrttable[⟨options⟩]}
```
8. Defining and Displaying Glossaries

The table style should not be set with the \texttt{style} package option, \texttt{\setglossarystyle} or the \texttt{style} option, as it’s only intended for use within \texttt{\printunsrttable}, which sets up the appropriate hooks to allow the style to work with \texttt{\printunsrtglossary} (which is used implicitly).

Tabular styles such as \texttt{long} create a longtable with one entry per row and no caption. The longheader style is similar but adds a header row, and the long-booktabs style includes rules above and below the header row and at the end of the table. In all these longtable styles, the glossary title is outside of the style, and is typically put in a sectioning command. Similarly, the glossary preamble \texttt{\glossarypreamble} and postamble \texttt{\glossarypostamble} are outside of longtable.

The table style, on the other hand, allows multiple entries per row. The glossary title (\texttt{title}) is the table caption with what’s normally the table of contents title (\texttt{toctitle}) as the caption title for the list of tables. Similarly, the preamble and postamble are included in the table header and footer, instead of being outside of the table.

This means that \texttt{\glossarysection}, \texttt{\glossarypreamble} and \texttt{\glossarypostamble} are outside of \texttt{\printunsrttable} to do nothing so that they aren’t shown outside of the longtable by \texttt{\printunsrtglossary}, otherwise there would be a duplication of the information in the header and footer of the table. The \texttt{\printunsrtglossary} hooks are used to insert the inter-block tabulation (\texttt{&}) character and new row command in the construction performed outside of longtable, which would otherwise cause issues if used directly in the table style.

The block styles (see §8.7.4.3) alter the way the table style sets up the longtable environment and the way that the entries are formatted. The top level glossary style command \texttt{\glossentry} is defined to do the block according to the designated block style, which includes the child entries, if the \texttt{childcount} field has been set and is non-zero.

The \texttt{\subglossentry} command is redefined to do nothing, but it won’t be used as the child entries are all filtered out. If you don’t use the \texttt{save-child-count} resource option, no child entries will be shown. There’s no recursive descent down the hierarchical levels.

This means that the child entries will be listed in one of the columns in the block, according to the style. This can make the column quite wide. The child names aren’t displayed by default but the block styles support the \texttt{subentrycounter} option. The child entries are listed in a tabular environment, which means they are contained in the same row as their parent and can’t be broken across a page.

A “block” indicates a block of columns used to format one entry (and, optionally, its children). One row of the table may contain multiple blocks. For example, a block may consist of two columns with the name in the first column and the description in the second, or may consist of three columns with the name in the first column, the symbol in the second, and the description in the third. So if a block style has 3 columns, and the desired number of blocks is set to 2, then the table will have a total of $3 \times 2 = 6$ columns.

The style supports up to 1 hierarchical level, but you will need the \texttt{save-child-count}
resource option if you want the level 1 sub-entries to show. Deeper level entries are omitted. Sub-entries are automatically filtered by a custom hook that \printunsrtglossaryentry- processhook is assigned to within \printunsrttable. This custom hook allows additional filtering to be employed with the command:

\glstableiffilter\{⟨entry-label⟩\}{⟨true⟩}{⟨false⟩}

This command should do ⟨true⟩ if the entry identified by ⟨entry-label⟩ should be skipped, otherwise it should do ⟨false⟩. The default definition simply does ⟨false⟩.

For example, the following will filter entries that have the category set to general:

\renewcommand{\glstableiffilter}[1]\
glsifcategory{#1}{general}

Note that if this command is redefined to do neither ⟨true⟩ nor ⟨false⟩ or does both, it will interfere with the width calculations if par isn’t set to the default par=false.

You can use the init option to locally redefine commands within \printunsrttable. For example:

\printunsrttable[init={\renewcommand{\glstableiffilter}[1]\
glsifcategory{##1}{general}%}]

An extra field (the “other” field) may be added with the other key. If this value is empty, then no extra field will be added. Some block styles, such as other-name and symbol-other put the other field in its own column. If the other field isn’t set, this will lead to an empty column.

If there isn’t a designated column for the other field, then block styles that show the description will put the other field in before the description, but in the same column as the description. Otherwise, block styles that don’t show the description, will put the other field after the name, but in the same column as the name.

The following example uses the name block style, which only has one column per block. The name is followed by the description in parentheses (if set), which is then followed by the child list. I redefined \glstableNameFmt to make the name appear in bold, to highlight it. I’ve used the par=ragged option, otherwise the table will be too wide to fit the page.

\usepackage[record,stylemods=table,subentrycounter]
This creates a table with two entries per row.
8. Defining and Displaying Glossaries

Example 150: Two entries per row with \printunsrttable

Table 1: Glossary

<table>
<thead>
<tr>
<th>Summary</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>class (aptent taciti)</td>
<td>consectetuer (mauris)</td>
</tr>
<tr>
<td>1) ad litora</td>
<td>1) leo justo</td>
</tr>
<tr>
<td>2) sociosqu</td>
<td>2) quis egestas</td>
</tr>
<tr>
<td>3) torquent per conubia</td>
<td></td>
</tr>
<tr>
<td>cum sociis (natoque penatibus)</td>
<td>eleifend (sit amet faucibus)</td>
</tr>
<tr>
<td>1) dis parturient montes</td>
<td>1) elementum</td>
</tr>
<tr>
<td>2) et magnis</td>
<td>2) urna sapien</td>
</tr>
<tr>
<td>mauris (libero eros)</td>
<td>non risus (morbi non felis)</td>
</tr>
<tr>
<td>1) dapibus porttitor, pede</td>
<td>1) ac libero</td>
</tr>
<tr>
<td>2) lacinia non</td>
<td>2) vulputate fringilla</td>
</tr>
<tr>
<td>3) sodales quis</td>
<td></td>
</tr>
<tr>
<td>nostra (per inceptos hymenaeos)</td>
<td>scelerisque (at)</td>
</tr>
<tr>
<td>1) mauris condimentum nulla</td>
<td>1) sem leo</td>
</tr>
<tr>
<td>2) morbi dapibus</td>
<td>2) tellus et tortor</td>
</tr>
<tr>
<td></td>
<td>3) nec, enim</td>
</tr>
<tr>
<td></td>
<td>4) sit amet</td>
</tr>
<tr>
<td></td>
<td>5) vehicula pellentesque</td>
</tr>
<tr>
<td></td>
<td>6) facilisis id</td>
</tr>
<tr>
<td></td>
<td>7) eu, nulla</td>
</tr>
<tr>
<td>ut tempor (laoreet quam)</td>
<td></td>
</tr>
<tr>
<td>1) wisi a libero</td>
<td></td>
</tr>
<tr>
<td>2) rutrum ut</td>
<td></td>
</tr>
<tr>
<td>3) semper</td>
<td></td>
</tr>
</tbody>
</table>

Note that each row is as deep as the entry with the most children. So where a row has one column with two children and another with seven, the row is deep enough to accommodate the seven child entries, which leaves a gap below the smaller list of two children.

8.7.4.1. Child Entries

Entries with a hierarchical level greater than 0 are filtered out (see above). This takes the leveloffset option into account. Child entries can be included, but only by checking if the
childcount field has been set and is non-zero. This is done by:

\glstableChildEntries{(entry-label)}

Note that \glstableiffilter filters top-level entries, and their child entries will also be filtered. Child entries for non-filtered top-level entries can be filtered by redefining:

\glstableiffilterchild{(entry-label)}{(true)}{(false)}

where \textit{(entry-label)} is the child entry label. This command should do \textit{(true)} if the child entry should be filtered and \textit{(false)} otherwise.

If the child count is non-zero, taking both childcount and child filtering into account, then \glstableChildEntries command will display the non-filtered children in the form:

\glstablePreChildren
\begin{glstablesubentries}
\glstableblocksubentry{(child-1-label)}
\glstableblocksubentry{(child-2-label)}
...
\glstableblocksubentrysep
\glstableblocksubentry{(child-n-label)}
\end{glstablesubentries}

This consists of the following.

\glstablePreChildren

Occurs at the start. If \texttt{par=justified} or \texttt{par=ragged}, this will do \texttt{par} otherwise it does nothing.

In general, it’s best not to list children with \texttt{par=false}, except with a style like \texttt{name} or \texttt{name-desc} with no description, as the table can end up too wide for the page.
This environment encapsulates the child list. By default, this does:

\begin{tabular}[t]{⟨align⟩}
⟨content⟩
\end{tabular}

The ⟨align⟩ argument is obtained by expanding:

\glstablesubentryalign

which takes the par setting into account.

Each child item is display using \glstableblocksubentry which is redefined by the block style.

The separator between each child item is given by:

\glstableblocksubentrysep

This just expands to \glstablenewline.

**8.7.4.2. Options**

The optional argument of \printunsrttable may have the options that can typically be passed to \printunsrtglossary, except that the nonumberlist and style options won’t have an effect. If you want the location list, it can simply be obtained from the location field in the appropriate style hook.

Some default settings are changed: groups=false and nogroupskiptrue. If you want letter group headings, you will need to both add groups=true to the options list and invoke \texttt{bib2gls} with the \texttt{--group} switch. The group headings will span the entire width of the table. This may result in empty blocks at the end of the previous row. If you want a vertical gap before the group heading (but not before the first group), you will need to add nogroupskipfalse, but you will also need to load glossary-longbooktabs. Note that this option is designed to be used with group headings and will have no effect with groups=false.

Additionally, the following options may also be used.

\texttt{blocks=(n)}

The value is the number of blocks in the table. The total number of columns in the table will be this value multiplied by the number of columns per block, which is determined by the block style. For example, the name-desc block style has two columns, so if there are three
blocks then there will be a total of six columns.

```
caption={boolean} default: true; initial: true
```

A boolean option that determines whether or not to include a caption. The caption on the first page of the table is produced with:

```
\glstablecaption{⟨lot title⟩}{⟨title⟩}{⟨label code⟩}
```

where ⟨label code⟩ is the code to create the label, if one has been supplied (either by an option such as numberedsection=autolabel or by the label option). The ⟨title⟩ argument will be \glossarytitle, which can be changed with the title option. The ⟨lot title⟩ argument is the title for the list of tables and is actually what would normally be the title for the table of contents, which can be set with the toctitle option. The default definition simply does:

```
\caption[⟨lot title⟩]{⟨label code⟩⟨title⟩}
```

An empty ⟨lot title⟩ (toctitle={}) will prevent the caption from being added to the list of tables.

---

The numberedsection option will only influence the label, not the table numbering. If you don’t want the table numbered, redefine \glstablecaption to use \caption*.

If the table spans across multiple pages, the caption for subsequent pages will be produced with:

```
\glstabelnextcaption{⟨lot title⟩}{⟨title⟩}
```

This ignores the ⟨lot title⟩ argument by default and does:

```
\caption[]{⟨title⟩}\glstablepostnextcaption
```

This has an empty optional argument to prevent the caption from being repeatedly added to the list of tables. The title is followed by:

```
\glstablepostnextcaption initial: Cont./
```

You can either redefine this command to adjust the content after the title or redefine \gls-
8. Defining and Displaying Glossaries

tablenextcaption, as appropriate.

\begin{table}
\begin{tabular}{|l|l|}
\hline
header=\{boolean\} & \textit{default: true; initial: true} \\
\hline
\end{tabular}
\end{table}

A boolean option to determine whether or not to show the header row. Note that a header with three column block styles, such as name-symbol-desc, can result in overfull lines. You may need to shorten the header text to fit.

The header text is produced with one of the following commands:

\begin{itemize}
\item \texttt{\textbackslash{}glstablenameheader}
\end{itemize}

Expands to the header for the name column. Just uses \texttt{\entryname} by default.

\begin{itemize}
\item \texttt{\textbackslash{}glstabledescheader}
\end{itemize}

Expands to the header for the description column for block styles like name-desc and name-symbol-desc. Just uses \texttt{\descriptionname} by default.

\begin{itemize}
\item \texttt{\textbackslash{}glstablesymbolheader}
\end{itemize}

Expands to the header for the symbol column for block styles like name-symbol and name-symbol-desc. Just uses \texttt{\symbolname} by default.

\begin{itemize}
\item \texttt{\textbackslash{}glstableotherheader}
\end{itemize}

Expands to the header for the other column. The default definition applies \texttt{\MFUsentencecase} to the other field label.

\begin{table}
\begin{tabular}{|l|l|}
\hline
rules=\{boolean\} & \textit{default: true; initial: true} \\
\hline
\end{tabular}
\end{table}

A boolean option to determine whether or not to show the horizontal rules (provided by booktabs). If used with header=\texttt{true}, there will be a rule above and below the header row. If used with header=\texttt{false}, there will only be one rule at the top of the table. In both cases, there will be a rule at the bottom of the table.

\begin{table}
\begin{tabular}{|l|}
\hline
blocksep=\{alignment spec\} & \textit{initial: |} \\
\hline
\end{tabular}
\end{table}
8. Defining and Displaying Glossaries

The value is inserted into the alignment specifier list between blocks. For example, the default value of the pipe character will insert a vertical line. Set this value to empty to remove it.

\texttt{par=⟨value⟩} \quad \textit{initial: false}

Indicates whether or not the columns should be paragraphs. The value may be one of: false, justified or ragged. The default \texttt{par=false} will just use one of the column specifiers l, r or c. The other values will use the p specifier, in which case the column widths will be calculated.

\texttt{other=⟨field-label⟩} \quad \textit{initial: empty}

This should be set to the internal field label of the other field or to empty if no other field should be included.

\texttt{init=⟨(code)⟩} \quad \textit{initial: empty}

The \texttt{⟨code⟩} will be added shortly before \texttt{\printunsrtglossary} is called and any local changes will be scoped.

\texttt{block-style=⟨value⟩} \quad \textit{initial: name-desc}

The block style. Available styles are listed in §8.7.4.3.

8.7.4.3. Block Styles

The block style may be set with the \texttt{block-style} option or with:

\texttt{\glstablesetstyle{⟨style-name⟩}}

The block styles are still under development, so the underlying commands are not yet documented and liable to change.

The following block styles are predefined.

\texttt{block-style=name}
8. Defining and Displaying Glossaries

Blocks have one column with the name, which is followed by the symbol and the description, if they have been set, in parentheses. The child list follows at the end of the column (if `childcount` is set and non-zero).

<table>
<thead>
<tr>
<th>block-style</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-desc</td>
<td>Block</td>
<td>This is the default style. Blocks have two columns with the name in the first column of the block and the description in the second. If the other field is set, it will follow the description. The child entries will be at the end of the second column (if <code>childcount</code> is set and non-zero).</td>
</tr>
<tr>
<td>desc-name</td>
<td>Block</td>
<td>As <code>name-desc</code> but with the columns swapped. The child entries (if <code>childcount</code> is set and non-zero) will be at the end of the first column.</td>
</tr>
<tr>
<td>name-symbol</td>
<td>Block</td>
<td>Blocks have two columns with the name in the first column of the block and the symbol in the second. If the other field is set, it will be placed after the name in the first column. The child entries are at the end of the first column (if <code>childcount</code> is set and non-zero).</td>
</tr>
<tr>
<td>symbol-name</td>
<td>Block</td>
<td>As <code>name-symbol</code> but with the columns swapped. The child entries (if <code>childcount</code> is set and non-zero) will be at the end of the second column.</td>
</tr>
<tr>
<td>name-other</td>
<td>Block</td>
<td>This is like <code>name-desc</code> but puts the other field in the second column. The description and symbol aren't shown. The child entries (if <code>childcount</code> is set and non-zero) will be at the end of the second column.</td>
</tr>
<tr>
<td>other-name</td>
<td>Block</td>
<td>This is like <code>desc-name</code> but puts the other field in the second column. The description and symbol aren't shown. The child entries (if <code>childcount</code> is set and non-zero) will be at the end of the second column.</td>
</tr>
</tbody>
</table>
8. Defining and Displaying Glossaries

end of the first column.

```
block-style=symbol-other
```

This is like `name-other` but shows the symbol instead of the name. The child entries (if `childcount` is set and non-zero) will be at the end of the second column.

```
block-style=other-symbol
```

This is like `other-name` but shows the symbol instead of the name. The child entries (if `childcount` is set and non-zero) will be at the end of the first column.

```
block-style=name-symbol-desc
```

Blocks have three columns with the name in the first column of the block, the symbol in the second, and the description in the third, preceded by the other field, if set. The child entries are at the end of the third column (if `childcount` is set and non-zero).

```
block-style=name-desc-symbol
```

Blocks have three columns with the name in the first column of the block, the description in the second, preceded by the other field, if set, and the symbol in the third. The child entries are at the end of the second column (if `childcount` is set and non-zero).

```
block-style=name-other-desc
```

Blocks have three columns with the name in the first column of the block, the other in the second, and the description in the third. The child entries are at the end of the third column (if `childcount` is set and non-zero).

```
block-style=desc-other-name
```

Blocks have three columns with the description in the first column of the block, the other in the second, and the name in the third. The child entries are at the end of the first column (if `childcount` is set and non-zero).

```
block-style=name-symbol-other-desc
```

493
8. Defining and Displaying Glossaries

Blocks have four columns with the name in the first column of the block, the symbol in the second, the other in the third, and the description in the fourth. The child entries are at the end of the fourth column (if \texttt{childcount} is set and non-zero).

\begin{verbatim}
block-style=\texttt{name\text{-}other\text{-}symbol\text{-}desc}
\end{verbatim}

Blocks have four columns with the name in the first column of the block, the other in the second, the symbol in the third, and the description in the fourth. The child entries are at the end of the fourth column (if \texttt{childcount} is set and non-zero).

\begin{verbatim}
block-style=\texttt{desc\text{-}symbol\text{-}other\text{-}name}
\end{verbatim}

Blocks have four columns with the description in the first column of the block, the symbol in the second, the other in the third, and the name in the fourth. The child entries are at the end of the first column (if \texttt{childcount} is set and non-zero).

\begin{verbatim}
block-style=\texttt{desc\text{-}other\text{-}symbol\text{-}name}
\end{verbatim}

Blocks have four columns with the description in the first column of the block, the other in the second, the symbol in the third, and the name in the fourth. The child entries are at the end of the first column (if \texttt{childcount} is set and non-zero).

8.7.4.4. Associated Commands

The rows are separated with:

\begin{verbatim}
\glstablenewline
\end{verbatim}

This simply does \texttt{\tabularnewline} (not \textbackslash \ which has a different action in paragraph columns).

The following commands are used in the column specifier where a left, right or centred column is required, taking the \texttt{par} option into account. Note that with \texttt{par=justified}, the result will always be \texttt{p\{\langle width\rangle\}}, whereas with \texttt{par=ragged} the paragraph will be ragged right or ragged left or have centring applied.

\begin{verbatim}
\glstableleftalign{\langle width\rangle}
\end{verbatim}

Expands to \texttt{l} or \texttt{p\{\langle width\rangle\}} or >\texttt{\protect\raggedrightp\{\langle width\rangle\}}, depending on the \texttt{par} setting.
This command is used in the column specifier where a left-justified column is required.

\glablerightalign{⟨width⟩}

Expands to r or p{⟨width⟩} or >\protect\raggedleftp{⟨width⟩}, depending on the par setting.

This command is used in the column specifier where a right-justified column is required.

\glablecenteralign{⟨width⟩}

Expands to c or p{⟨width⟩} or >\protect\centeringp{⟨width⟩}, depending on the par setting.

This command is used in the column specifier where a centred column is required.

\glablenamemalign

Expands to the alignment for the name column. The default definition uses left alignment:

\glableleftalign{\glablenamewidth}

\glabledescmalign

Expands to the alignment for the description column. The default definition uses left alignment:

\glableleftalign{\glabledescwidth}

\glablesymbolmalign

Expands to the alignment for the symbol column, in block styles where the symbol has its own column. The default definition uses centred alignment:

\glablecenteralign{\glablesymbolwidth}
### Defining and Displaying Glossaries

<table>
<thead>
<tr>
<th>Environment</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>\glstableothercolalign</td>
<td>Expands to the alignment for the other column, in block styles where the other field has its own column. The default definition uses left alignment.</td>
</tr>
<tr>
<td></td>
<td>\glstableleftalign{\glstableotherwidth}</td>
<td>If \texttt{par=justified} or \texttt{par=ragged}, the column widths will be calculated. The following length registers will be set, where applicable to the block style.</td>
</tr>
<tr>
<td></td>
<td>\glstablenamewidth</td>
<td>The width of the name column.</td>
</tr>
<tr>
<td></td>
<td>\glstabledescwidth</td>
<td>The width of the description column.</td>
</tr>
<tr>
<td></td>
<td>\glstablesymbolwidth</td>
<td>The width of the symbol column.</td>
</tr>
<tr>
<td></td>
<td>\glstableotherwidth</td>
<td>The width of the other column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unless \texttt{par=false}, the table will be the width of a line and each block will have equal width.</td>
</tr>
<tr>
<td></td>
<td>\glstableblockwidth</td>
<td>Note that in all the above, the width doesn’t include the inter-column space given by \texttt{\tabcolsep}. The length registers below are initialise to 5pt, and can be redefined as appropriate.</td>
</tr>
<tr>
<td></td>
<td>\glstablepostpreambleskip</td>
<td>\textit{initial: 5pt}</td>
</tr>
</tbody>
</table>
The vertical skip after the preamble.

\glstableprepostambleskip \textit{initial: 5pt}

The vertical skip before the postamble.

Formatting for the name, symbol, description and other field values are applied by the following commands.

\glstableNameFmt\{⟨text⟩\}

Formatting applied to the name. Simply does ⟨text⟩ by default. Note that the argument ⟨text⟩ will \glossentryname\{⟨label⟩\}, so any formatting applied by that command will also be in effect.

\glstableSubNameFmt\{⟨text⟩\}

Formatting applied to the child name. Does nothing by default, which means that the child name won’t show.

\glstableSymbolFmt\{⟨text⟩\}

Formatting applied to the symbol. Simply does ⟨text⟩ by default. Note that the argument ⟨text⟩ will \glossentrysymbol\{⟨label⟩\}, so any formatting applied by that command will also be in effect.

\glstableSubSymbolFmt\{⟨text⟩\}

Formatting applied to the child symbol. Just does \glstableSymbolFmt by default.

\glstableDescFmt\{⟨text⟩\}

Formatting applied to the description. Simply does ⟨text⟩ by default. Note that the argument ⟨text⟩ will be:

\glossentrydesc\{⟨label⟩\}\glspostdescription

so any formatting applied by \glossentrydesc will also be in effect. Note that the post-
8. Defining and Displaying Glossaries

description hook is included in the formatted.

\glstableSubDescFmt{(text)}

Formatting applied to the child description. Just does \glstableDescFmt by default.

The other field’s internal label is provided by expanding:

\glstableotherfield

initial: empty

This is redefined by the other option, but it may be redefined before \printunsrttable if a default field is required.

\glstableOtherFmt{(text)}

The formatting applied to the other field. This just does \langle text \rangle by default. The field value itself is displayed with:

\glstableOther{\langle entry-label \rangle}

The default definition does:

\glstableOtherFmt{%
\glsxtrusefield{\langle entry-label \rangle}\{\glstableotherfield}\}

The value for the child entries is displayed with:

\glstableSubOther{\langle entry-label \rangle}

The default definition simply does \glstableOther{\langle entry-label \rangle}

You can test whether or not the other field is set for a given entry with:

\glstableifhasotherfield{\langle entry-label \rangle}{\langle true \rangle}{\langle false \rangle}

This does \langle true \rangle of the other field is non-void (according to \ifglsfieldvoid) otherwise it does \langle false \rangle. This will always do \langle false \rangle if \glstableotherfield is void.

498
The column headers are supplied by the following commands, where applicable.

\glstablenameheader

The header for the name column. The default definition is \entryname.

\glstabledescheader

The header for the description column. The default definition is \descriptionname.

\glstablesymbolheader

The header for the symbol column. The default definition is \symbolname.

\glstableotherheader

The header for the other column. The default definition is:

\textbf{\glstableotherfield}

The column headers are formatted according to:

\glstableHeaderFmt{⟨text⟩}

The default definition is \textbf{⟨text⟩}.

The remaining commands are undocumented as they are liable to change.
9. Accessibility Support

The glossaries package comes with a supplementary package glossaries-accsupp that helps provide accessibility support. The glossaries-extra package provides additional support, but only if the glossaries-accsupp package has already been loaded when the relevant commands are defined. The best and simplest way to do this is through the accsupp package option.

See the “Accessibility Support” chapter in the glossaries user guide for further information about glossaries-accsupp.

9.1. Abbreviations

The accessibility fields relating to abbreviations are shortaccess, shortpluralaccess, longaccess and longpluralaccess. These provide the replacement text for the corresponding short, shortplural, long and longplural fields. The access field provides the replacement text for the name field.

Some of these accessibility fields are automatically assigned by \newabbreviation if they haven’t been set.

\glsxtrassignactualsetup

This command is used to locally redefine common formatting commands so that they can be stripped to obtain only the text. You can add additional commands with \appto. For example, the following eccentric example has some strange styling in the abbreviation:

\newabbreviation{foo}{f\textsuperscript{o}\textsubscript{o}}{furry old otters}

If an accessibility field is being automatically assigned with text obtained from the short value, then the subscript and superscript commands will need to be stripped. These need to be locally redefined to just do their arguments:

\appto\glsxtrassignactualsetup{%
  \letcs{\textsuperscript}{@firstofone}%
  \letcs{\textsubscript}{@firstofone}%
}
The attributes that specifically relate to accessibility in abbreviations are listed below. The “actual short value” means the value obtained from the short value after any markup commands have have locally redefined using \glxtrassignactualsetup. The actual short value may then be modified by these attributes. Similarly, for the “actual long value”.

Finally, if shortaccess hasn’t already been set, it will be set to:

\glsdefaultshortaccess{(actual long)}{(actual short)}

(with \glsdefaultshortaccess expanded). This command is provided by glossaries-accsupp and is defined to do just \{(actual long)\}. It was redefined by glossaries-extra v1.42 to do \{(actual long)\} \{(actual short)\}, but has been reverted back to its original definition in v1.49.

accessinsertdots={boolean}

If the shortaccess key hasn’t been set then this attribute will be checked. If true, the actual short value will have dots inserted (as per insertdots). Note that if this attribute hasn’t been set but insertdots is true (and the shortaccess key hasn’t been set), then the actual short value will also have dots inserted.

accessaposplural={boolean}

If the shortpluralaccess key hasn’t been set then this attribute will be checked. If true, the actual short plural value will have the apostrophe suffix (similar to aposplural but using \glxtrabbrvpluralsuffix instead of \abbrvpluralsuffix). Note that if this attribute hasn’t been set but aposplural is true (and the shortpluralaccess key hasn’t been set), then the actual short plural value will also have the apostrophe suffix.

accessnoshortplural={boolean}

If the shortpluralaccess key hasn’t been set and the accessaposplural attribute hasn’t been set, then this attribute will be checked. If true, the actual short plural value will be the same as the singular (as noshortplural). Note that if this attribute hasn’t been set but noshortplural is true (and the shortpluralaccess key hasn’t been set), then the actual short plural value will also be the singular form.

nameshortaccess={boolean}

If the access key hasn’t been set and this attribute is true, then the access field will be set
9. Accessibility Support

to the same as the shortaccess.

\textshortaccess\langle boolean\rangle

If the textaccess key hasn’t been set and this attribute is true, then the textaccess field will be set to the same as the shortaccess. Additionally, if the pluralaccess key hasn’t been set, then it will be set to the same as the shortpluralaccess value.

\firstshortaccess\langle boolean\rangle

If the firstaccess key hasn’t been set and this attribute is true, then the firstaccess field will be set to the same as the shortaccess. Additionally, if the firstpluralaccess key hasn’t been set, then it will be set to the same as the shortpluralaccess value.

9.2. Accessibility Wrappers

The glossary style commands such as \glossentryname incorporate accessibility support by using the \glsaccess\langle field\rangle commands instead of the corresponding \glsentry\langle field\rangle commands.

If the glossaries-accsupp package hasn’t been loaded or if the relevant accessibility field hasn’t been set, these commands simply do the corresponding \glsentry\langle field\rangle command.

\\glsaccessname{\langle entry-label\rangle}

This shows the name field encapsulated with \glsnameaccessdisplay or just \glsentryname{\langle entry-label\rangle}.

\\Glsaccessname{\langle entry-label\rangle}

As above but sentence case.

\GLSaccessname{\langle entry-label\rangle}

As above but all caps.

\\glsaccesstext{\langle entry-label\rangle}
9. Accessibility Support

This shows the text field encapsulated with \glistextaccessdisplay or just \glsentrytext{(entry-label)}.

\Glsaccess{\langle entry-label \rangle}

As above but sentence case.

\GLSaccess{\langle entry-label \rangle}

As above but all caps.

\glsaccessplural{\langle entry-label \rangle}

This shows the plural field encapsulated with \glsppluralaccessdisplay or just \glsentryplural{(entry-label)}.

\Glsaccessplural{\langle entry-label \rangle}

As above but sentence case.

\GLSaccessplural{\langle entry-label \rangle}

As above but all caps.

\glsaccessfirst{\langle entry-label \rangle}

This shows the first field encapsulated with \glsfirstaccessdisplay or just \glsentryfirst{(entry-label)}.

\Glsaccessfirst{\langle entry-label \rangle}

As above but sentence case.

\GLSaccessfirst{\langle entry-label \rangle}
9. Accessibility Support

As above but all caps.

\glsaccessfirstplural{⟨entry-label⟩}

This shows the \texttt{firstplural} field encapsulated with \texttt{\glsfirstpluralaccessdisplay} or just \texttt{\glsentryfirstplural{⟨entry-label⟩}}.

\Glsaccessfirstplural{⟨entry-label⟩}

As above but sentence case.

\GLSaccessfirstplural{⟨entry-label⟩}

As above but all caps.

\glsaccesssymbol{⟨entry-label⟩}

This shows the \texttt{symbol} field encapsulated with \texttt{\glssymbolaccessdisplay} or just \texttt{\glsentrysymbol{⟨entry-label⟩}}.

\Glsaccesssymbol{⟨entry-label⟩}

As above but sentence case.

\GLSaccesssymbol{⟨entry-label⟩}

As above but all caps.

\glsaccesssymbolplural{⟨entry-label⟩}

This shows the \texttt{symbolplural} field encapsulated with \texttt{\glssymbolpluralaccessdisplay} or just \texttt{\glsentrysymbolplural{⟨entry-label⟩}}.

\Glsaccesssymbolplural{⟨entry-label⟩}
9. Accessibility Support

As above but sentence case.

\GLSaccesssymbolplural{⟨entry-label⟩}

As above but all caps.

\glsaccessdesc{⟨entry-label⟩}

This shows the description field encapsulated with \glsdescriptionaccessdisplay or just \glsentrydesc{⟨entry-label⟩}.

\Glsaccessdesc{⟨entry-label⟩}

As above but sentence case.

\GLSaccessdesc{⟨entry-label⟩}

As above but all caps.

\glsaccessdescplural{⟨entry-label⟩}

This shows the descriptionplural field encapsulated with \glsdescriptionpluralaccessdisplay or just \glsentrydescplural{⟨entry-label⟩}.

\Glsaccessdescplural{⟨entry-label⟩}

As above but sentence case.

\GLSaccessdescplural{⟨entry-label⟩}

As above but all caps.

\glsaccessshort{⟨entry-label⟩}
9. Accessibility Support

This shows the short field encapsulated with \glsshortaccessdisplay or just \glsentryshort{⟨entry-label⟩}.

\Glsaccessshort{⟨entry-label⟩}

As above but sentence case.

\GLSaccessshort{⟨entry-label⟩}

As above but all caps.

\glsaccessshortpl{⟨entry-label⟩}

This shows the shortplural field encapsulated with \glsshortpluralaccessdisplay or just \glsentryshortpl{⟨entry-label⟩}.

\Glsaccessshortpl{⟨entry-label⟩}

As above but sentence case.

\GLSaccessshortpl{⟨entry-label⟩}

As above but all caps.

\glsaccesslong{⟨entry-label⟩}

This shows the long field encapsulated with \glslongaccessdisplay or just \glsentrylong{⟨entry-label⟩}.

\Glsaccesslong{⟨entry-label⟩}

As above but sentence case.

\GLSaccesslong{⟨entry-label⟩}
9. Accessibility Support

As above but all caps.

\glsaccesslongpl{⟨entry-label⟩}

This shows the *longplural* field encapsulated with \glslongpluralaccessdisplay or just \glsentrylongpl{⟨entry-label⟩}.

\GLSaccesslongpl{⟨entry-label⟩}

As above but sentence case.

\GLSaccesslongpl{⟨entry-label⟩}

As above but all caps.

\glsaccessuseri{⟨entry-label⟩}

This shows the *user1* field encapsulated with \glsuseriaccessdisplay or just \glsentryuseri{⟨entry-label⟩}.

\GLSaccessuseri{⟨entry-label⟩}

As above but sentence case.

\GLSaccessuseri{⟨entry-label⟩}

As above but all caps.

\glsaccessuserii{⟨entry-label⟩}

This shows the *user2* field encapsulated with \glsuseriiaccessdisplay or just \glsentryuserii{⟨entry-label⟩}.

\GLSaccessuserii{⟨entry-label⟩}
9. Accessibility Support

As above but sentence case.

\GLSaccessuserii\{\langle\text{entry-label}\rangle\}

As above but all caps.

\glsaccessuserii\{\langle\text{entry-label}\rangle\}

This shows the user3 field encapsulated with \glsuseriiaccessdisplay or just \glsentryuserii\{\langle\text{entry-label}\rangle\}.

\GLSaccessuserii\{\langle\text{entry-label}\rangle\}

As above but sentence case.

\Glsaccessuserii\{\langle\text{entry-label}\rangle\}

As above but all caps.

\glsaccessuseriv\{\langle\text{entry-label}\rangle\}

This shows the user4 field encapsulated with \glsuserivaccessdisplay or just \glsentryuseriv\{\langle\text{entry-label}\rangle\}.

\Glsaccessuseriv\{\langle\text{entry-label}\rangle\}

As above but sentence case.

\GLSaccessuseriv\{\langle\text{entry-label}\rangle\}

As above but all caps.

\glsaccessuserv\{\langle\text{entry-label}\rangle\}
9. Accessibility Support

This shows the user5 field encapsulated with \glsuseraccessdisplay or just \glsentryuserv\{(entry-label)\}.

\Glsaccessuserv\{(entry-label)\}

As above but sentence case.

\GLSaccessuserv\{(entry-label)\}

As above but all caps.

\glsaccessuservi\{(entry-label)\}

This shows the user6 field encapsulated with \glsuserviaccessdisplay or just \glsentryuservi\{(entry-label)\}.

\Glsaccessuservi\{(entry-label)\}

As above but sentence case.

\GLSaccessuservi\{(entry-label)\}

As above but all caps.

9.3. Inner Formatting Wrappers

These \glsaccessfmt\{field\} commands are similar to the corresponding \glsaccess\{field\} commands described above, but they format the field value using \glsfmtfield, \Glsfmtfield or \GLSfmtfield with the supplied \{cs\} encapsulating command.

The default entry display style \glsgenentryfmt, and the predefined abbreviation styles all incorporate accessibility support by using these commands in order to support the inner formatting.

\glsaccessfmtname\{(insert)\}\{(cs)\}\{(entry-label)\}
9. Accessibility Support

This shows the name field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glsnameaccessdisplay.

\Glsaccessfmtname{(insert)}{(cs)}{(entry-label)}

As above but sentence case.

\GLSaccessfmtname{(insert)}{(cs)}{(entry-label)}

As above but all caps.

\glsaccessfmttext{(insert)}{(cs)}{(entry-label)}

This shows the text field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glstextaccessdisplay.

\Glsaccessfmttext{(insert)}{(cs)}{(entry-label)}

As above but sentence case.

\GLSaccessfmttext{(insert)}{(cs)}{(entry-label)}

As above but all caps.

\glsaccessfmtplural{(insert)}{(cs)}{(entry-label)}

This shows the plural field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glspluralaccessdisplay.

\Glsaccessfmtplural{(insert)}{(cs)}{(entry-label)}

As above but sentence case.

\GLSaccessfmtplural{(insert)}{(cs)}{(entry-label)}
9. Accessibility Support

As above but all caps.

\glsaccessfmtfirst\{(⟨insert⟩)⟨(cs)⟩{(entry-label)}\}

This shows the \textit{first} field formatted with ⟨(cs)⟩ and, if accessibility support provided, encapsulated with \glsfirstaccessdisplay.

\GLSaccessfmtfirst\{(⟨insert⟩)⟨(cs)⟩{(entry-label)}\}

As above but sentence case.

\GLSaccessfmtfirstplural\{(⟨insert⟩)⟨(cs)⟩{(entry-label)}\}

As above but all caps.

\glsaccessfmtfirstplural\{(⟨insert⟩)⟨(cs)⟩{(entry-label)}\}

This shows the \textit{firstplural} field formatted with ⟨(cs)⟩ and, if accessibility support provided, encapsulated with \glsfirstpluralaccessdisplay.

\GLSaccessfmtfirstplural\{(⟨insert⟩)⟨(cs)⟩{(entry-label)}\}

As above but sentence case.

\GLSaccessfmtsymbol\{(⟨insert⟩)⟨(cs)⟩{(entry-label)}\}

As above but all caps.

\glsaccessfmtsymbol\{(⟨insert⟩)⟨(cs)⟩{(entry-label)}\}

This shows the \textit{symbol} field formatted with ⟨(cs)⟩ and, if accessibility support provided, encapsulated with \glssymbolaccessdisplay.

\GLSaccessfmtsymbol\{(⟨insert⟩)⟨(cs)⟩{(entry-label)}\}
9. Accessibility Support

As above but sentence case.

\GLSaccessfmtsymbol\{\textit{insert}\}\{\textit{cs}\}\{\textit{entry-label}\}

As above but all caps.

\glsaccessfmtsymbolplural\{\textit{insert}\}\{\textit{cs}\}\{\textit{entry-label}\}

This shows the symbolplural field formatted with \textit{cs} and, if accessibility support provided, encapsulated with \texttt{\glssymbolpluralaccessdisplay}.

\GLSaccessfmtsymbolplural\{\textit{insert}\}\{\textit{cs}\}\{\textit{entry-label}\}

As above but sentence case.

\GLSaccessfmtsymbolplural\{\textit{insert}\}\{\textit{cs}\}\{\textit{entry-label}\}

As above but all caps.

\glsaccessfmtdesc\{\textit{insert}\}\{\textit{cs}\}\{\textit{entry-label}\}

This shows the description field formatted with \textit{cs} and, if accessibility support provided, encapsulated with \texttt{\glsdescriptionaccessdisplay}.

\GLSaccessfmtdesc\{\textit{insert}\}\{\textit{cs}\}\{\textit{entry-label}\}

As above but sentence case.

\GLSaccessfmtdesc\{\textit{insert}\}\{\textit{cs}\}\{\textit{entry-label}\}

As above but all caps.

\glsaccessfmtdescplural\{\textit{insert}\}\{\textit{cs}\}\{\textit{entry-label}\}

This shows the descriptionplural field formatted with \textit{cs} and, if accessibility support provided, encapsulated with \texttt{\glsdescriptionpluralaccessdisplay}.

\GLSaccessfmtdescplural\{\textit{insert}\}\{\textit{cs}\}\{\textit{entry-label}\}
9. Accessibility Support

provided, encapsulated with \glsdescriptionpluralaccessdisplay.

\Glsaccessfmtdescplural{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but sentence case.

\GLSaccessfmtdescplural{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but all caps.

\glsaccessfmtshort{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

This shows the short field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glsshortaccessdisplay.

\Glsaccessfmtshort{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but sentence case.

\GLSaccessfmtshort{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but all caps.

\glsaccessfmtshortpl{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

This shows the shortplural field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glsshortpluralaccessdisplay.

\Glsaccessfmtshortpl{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but sentence case.

\GLSaccessfmtshortpl{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
9. Accessibility Support

As above but all caps.

\glsaccessfmtlong{(insert)\{cs\}\{entry-label\}}

This shows the long field formatted with \langle cs \rangle and, if accessibility support provided, encapsulated with \glslongaccessdisplay.

\GLSaccessfmtlong{(insert)\{cs\}\{entry-label\}}

As above but sentence case.

\GLSaccessfmtlongpl{(insert)\{cs\}\{entry-label\}}

This shows the longplural field formatted with \langle cs \rangle and, if accessibility support provided, encapsulated with \glslongpluralaccessdisplay.

\GLSaccessfmtuseri{(insert)\{cs\}\{entry-label\}}

This shows the user1 field formatted with \langle cs \rangle and, if accessibility support provided, encapsulated with \glsuseriaccessdisplay.
9. Accessibility Support

As above but sentence case.

\GLSaccessfmtuseri{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but all caps.

\glsaccessfmtuserii{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

This shows the user2 field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glsuseriiaccessdisplay.

\Glsaccessfmtuserii{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but sentence case.

\GLSaccessfmtuseri{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but all caps.

\glsaccessfmtuseriii{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

This shows the user3 field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glsuseriiiaccessdisplay.

\Glsaccessfmtuseriii{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but sentence case.

\GLSaccessfmtuseriii{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but all caps.

\glsaccessfmtuseriv{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
This shows the user4 field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glsuserivaccessdisplay.

\Glsaccessfmtuseriv{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but sentence case.

\GLSaccessfmtuseriv{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but all caps.

\glsaccessfmtuserv{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

This shows the user5 field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glsuservaccessdisplay.

\Glsaccessfmtuserv{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but sentence case.

\GLSaccessfmtuserv{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but all caps.

\glsaccessfmtuservi{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

This shows the user6 field formatted with ⟨cs⟩ and, if accessibility support provided, encapsulated with \glsuserviaccessdisplay.

\Glsaccessfmtuservi{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but sentence case.

\GLSaccessfmtuservi{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}

As above but all caps.

516
10. Categories

For multi-entry categories, see §7.8.

Each entry defined by `\newglossaryentry` (or commands that internally use it such as `\newabbreviation`) is assigned a category through the `category` key. You may add any category that you like, but since the category is a label used in the creation of some control sequences, avoid problematic characters within the category label. (So take care if you have babel shorthands on that make some characters active.)

The use of categories can give you more control over the way entries are displayed in the text or glossary. Note that an entry’s category is independent of the glossary type. Be careful not to confuse `category` with `type`.

\glscategory{⟨entry-label⟩}

Expands to the category of the given entry or does nothing if the entry doesn’t exist (analogous to `\glsentryname`).

\glsifcategory{⟨entry-label⟩}{⟨category⟩}{⟨true⟩}{⟨false⟩}

Tests if the entry given by `⟨entry-label⟩` has the `category` set to `⟨category⟩`.

An entry may have its `category` field changed using commands such as `\GlsXtrSetField` (see §3.5). In addition, the following commands are provided to batch set the `category` for a collection of entries.

\glsxtrsetcategory{⟨entry-labels⟩}{⟨category-label⟩}

Globally sets the `category` field to the fully expanded `⟨category-label⟩` for each entry listed in `⟨entry-labels⟩`.

\glsxtrsetcategoryforall{⟨glossary-labels⟩}{⟨category-label⟩}
Globally sets the \texttt{category} field to the fully expanded \langle\texttt{category-label}\rangle for each entry belonging to the glossaries listed in \langle\texttt{glossary-labels}\rangle.

There are also some iterative commands available:

\begin{verbatim}
\glsforeachincategory[\langle\texttt{glossary-types}\rangle]{\langle\texttt{category}\rangle}{\langle\texttt{category}\rangle}{\langle\texttt{glossary-cs}\rangle}{\langle\texttt{label-cs}\rangle}{\langle\texttt{body}\rangle}
\end{verbatim}

This iterates through all entries in the glossaries identified by the comma-separated list \langle\texttt{glossary-labels}\rangle that have the category given by \langle\texttt{category-label}\rangle and performs \langle\texttt{body}\rangle for each match. Within \langle\texttt{body}\rangle, you can use \langle\texttt{glossary-cs}\rangle and \langle\texttt{label-cs}\rangle (which must be control sequences) to access the current glossary and entry label. If \langle\texttt{glossary-labels}\rangle is omitted, all glossaries are assumed.

\begin{verbatim}
\glsforeachwithattribute[\langle\texttt{glossary-types}\rangle]{\langle\texttt{attribute-label}\rangle}{\langle\texttt{attribute-value}\rangle}{\langle\texttt{glossary-cs}\rangle}{\langle\texttt{label-cs}\rangle}{\langle\texttt{body}\rangle}
\end{verbatim}

This iterates over all entries in the given list of glossaries that have a category with the given \langle\texttt{attribute-label}\rangle set to \langle\texttt{attribute-value}\rangle and performs \langle\texttt{body}\rangle at each iteration. If \langle\texttt{glossary-types}\rangle is omitted, the list of all non-ignored glossaries is assumed. The remaining arguments are as for \texttt{\glsforeachincategory}.

### 10.1. Known Categories

These are the category labels that are set or referenced by glossaries-extra.

- \texttt{general}
  
  The default category assumed by \texttt{\newglossaryentry}.

- \texttt{abbreviation}
  
  The default category assumed by \texttt{\newabbreviation}.

- \texttt{acronym}
  
  The default category assumed by \texttt{\newacronym}.

- \texttt{index}
10. Categories

The default category assumed by \texttt{\newterm}.

\begin{itemize}
\item \texttt{symbol}\index{symbol}
\end{itemize}

The default category assumed by \texttt{\glsxtrnewsymbol}.

\begin{itemize}
\item \texttt{number}\index{number}
\end{itemize}

The default category assumed by \texttt{\glsxtrnewnumber}.

\section*{10.2. Attributes}

Each category may have a set of attributes, where each attribute has an associated value for its given category. An entry’s attribute set corresponds to the attributes associated with the entry’s category.

As with the category, the attribute name is also a label. You can provide your own custom attributes, which you can set and access with the commands described in §10.2.2.

\subsection*{10.2.1. Known Attributes}

This section lists attributes that glossaries-extra sets or accesses. If an attribute hasn’t been set, a default is assumed. For boolean attributes, the test may simply be to determine if the attribute has been set to \texttt{true}, in which case any other value or a missing value will be interpreted as false. Conversely, the test may be to determine if the attribute has been set to \texttt{false}, in which case any other value or a missing value will be interpreted as true.

See §7.8 for attributes relating to multi-entry categories.

\subsection*{10.2.1.1. Abbreviation Attributes}

See §9.1 for abbreviation accessibility attributes.

\begin{itemize}
\item \texttt{regular=\{boolean\}}\index{regular}
\end{itemize}

This attribute indicates whether or not an entry should be considered a regular entry. This enables \texttt{\glsgenentryfmt} to determine whether to use \texttt{\glsgenentryfmt} or \texttt{\glsxtrgenabbrvfmt}. 

519
10. Categories

The \texttt{general} and \texttt{acronym} categories have the \texttt{regular} attribute automatically set to \texttt{true}. Some abbreviation styles change this value.

\begin{verbatim}
discardperiod=(boolean)
\end{verbatim}

If set to “true”, the post-link hook will discard a full stop that follows non-plural commands like \texttt{\gls} or \texttt{\glstext} (see §5.5.4).

This attribute doesn’t apply to the accessibility fields. See §9.1 for attributes related to accessibility support for abbreviations.

Note that this can cause a problem if you access a field that doesn’t end with a full stop. For example:

\begin{verbatim}
\newabbreviation [user1=\{German Speaking \TeX\ User Group\}] {dante} {\{Deutschsprachige Anwendervereinigung \TeX\ e.V.\}}
\end{verbatim}

Here the \texttt{short} and \texttt{long} fields end with a full stop, but the \texttt{user1} field doesn’t. The simplest solution in this situation is to put the sentence terminator in the final optional argument. For example:

\begin{verbatim}
\glsuseri{dante}[.]
\end{verbatim}

This will bring the punctuation character inside the link text and it won’t be discarded.

\begin{verbatim}
pluraldiscardperiod=(boolean)
\end{verbatim}

If this attribute is set to “true” and the \texttt{discardperiod} attribute is set to “true”, this will behave as above for the plural commands like \texttt{\glspl} or \texttt{\glsplural}.

\begin{verbatim}
retainfirstuseperiod=(boolean)
\end{verbatim}

If this attribute is set to “true” then the discard is determined by \texttt{\glsxtrdiscardperiod-retainfirstuse}, regardless of the \texttt{discardperiod} or \texttt{pluraldiscardperiod} attributes. This is useful for \texttt{\langle short \rangle (\langle long \rangle)} abbreviation styles where only the short form has a trailing full stop.
This attribute doesn’t apply to the accessibility fields. See §9.1 for attributes related to accessibility support for abbreviations.

\texttt{markwords=\langle boolean\rangle}

If this attribute is set to “true” any entry defined using \texttt{\newabbreviation} will automatically have spaces in the long form replaced with:

\texttt{\glsxtrwordsep}

and each word is encapsulated with:

\texttt{\glsxtrword{\langle word\rangle}}

For example:

\begin{verbatim}
\glssetcategoryattribute{abbreviation}{markwords}{true}
\newabbreviation{ip}{IP}{Internet Protocol}
\end{verbatim}

is essentially the same as

\begin{verbatim}
\newabbreviation{ip}{IP}{\glsxtrword{Internet}\glsxtrwordsep\glsxtrword{Protocol}}
\end{verbatim}

The “hyphen” styles, such as long-hyphen–short-hyphen, take advantage of this markup. If the inserted material (provided in the final argument of \texttt{\gls}-like commands) starts with a hyphen then \texttt{\glsxtrwordsep} is locally redefined to a hyphen. (The default value is a space). Note that this only applies to commands like \texttt{\gls} and not like \texttt{\glsxtrlong}. You can provide your own localised switch, if required. For example:

\begin{verbatim}
\newcommand{\hyplong}[2][]{%
  \def\glsxtrwordsep{-}\glsxtrlong[#1][#2]}
\end{verbatim}

This setting will also adjust the long plural. This attribute is only applicable to entries defined using \texttt{\newabbreviation} (or \texttt{\newacronym} if it’s using \texttt{\newabbreviation}.)
10. Categories

This setting may result in the \glsxtrword and \glsxtrwordsep markup ending up in the sort field, depending on the style in use.

\texttt{markshortwords=} \langle \text{boolean}\rangle

This is similar to \texttt{markwords} but applies to the short form. (Only useful for abbreviations that contain spaces.) This attribute is only applicable to entries defined using \newabbreviation (or \newacronym if it’s using \newabbreviation.)

This setting will only adjust the short plural if the shortplural key isn’t used. This setting will take precedence over insertdots.

This setting may result in the \glsxtrword and \glsxtrwordsep markup ending up in the sort field, depending on the style in use.

\texttt{insertdots=} \langle \text{boolean}\rangle

If this attribute is set to “true” any entry defined using \newabbreviation will automatically have full stops inserted after each letter. The entry will be defined with those dots present as though they had been present in the \langle short\rangle argument of \newabbreviation (rather than inserting them every time the entry is used). The short plural form defaults to the new dotted version of the original \langle short\rangle form with the plural suffix appended. This setting is incompatible with \texttt{markshortwords}. This attribute is only applicable to entries defined using \newabbreviation (or \newacronym if it’s using \newabbreviation.)

If you explicitly override the short plural using the shortplural key, you must explicitly insert the dots yourself (since there’s no way for the code to determine if the plural has a suffix that shouldn’t be followed by a dot).

This attribute is best used with the discardperiod attribute set to “true”.

\texttt{aposplural=} \langle \text{boolean}\rangle

If this attribute is set to “true”, \newabbreviation will insert an apostrophe (’') before the plural suffix for the short plural form (unless explicitly overridden with the shortplural key). The long plural form is unaffected by this setting. This setting overrides noshortplural. This attribute is only applicable to entries defined using \newabbreviation (or
10. Categories

\newacronym if it’s using \newabbreviation.) Check with your supervisor, publisher or editor if you want to use this attribute as this usage is controversial.

\begin{itemize}
  \item \texttt{noshortplural} = \langle boolean \rangle
  \end{itemize}

If this attribute is set to “true”, \newabbreviation won’t append the plural suffix for the short plural form. This means the \texttt{short} and \texttt{shortplural} values will be the same unless explicitly overridden. \textit{This setting is incompatible with aposplural}. This attribute is only applicable to entries defined using \newabbreviation (or \newacronym if it’s using \newabbreviation).

\begin{itemize}
  \item \texttt{tagging} = \langle boolean \rangle
  \end{itemize}

If this attribute is set to “true”, the tagging command defined by \GlsXtrEnableInitialTagging will be activated to use \glsxtrtagfont in the glossary (see §4.4)

10.2.1.2. Attributes that Alter \glslink Options

\begin{itemize}
  \item \texttt{nohyperfirst} = \langle boolean \rangle
  \end{itemize}

When used with the \gls-like commands, if this attribute is set to true, this will automatically suppress the hyperlink on first use.

This settings can be overridden by explicitly setting the \texttt{hyper} key on or off in the optional argument of the \gls-like command.

As from version 1.07, \glsfirst, \Glsfirst, \GLSfirst and their plural versions (which should ideally behave in a similar way to the first use of \gls or \glspl) now honour this attribute (but not the package-wide \texttt{hyperfirst=false} option, which matches the behaviour of glossaries). If you want commands like \glsfirst to ignore the \texttt{nohyperfirst} attribute then just redefine \glsxtrchecknohyperfirst to do nothing.

\begin{itemize}
  \item \texttt{nohypernext} = \langle boolean \rangle
  \end{itemize}

If set to true, this will automatically set \texttt{hyper=false} on subsequent use when using the
10. Categories

\gls-like commands.

\begin{itemize}
\item \texttt{nohyper}=$\langle$boolean$\rangle$
\end{itemize}

If set to \texttt{true}, this will automatically set \texttt{hyper=false} when using the \texttt{\gls} or \texttt{\glstext}-like commands.

\begin{itemize}
\item \texttt{indexonlyfirst}=$\langle$boolean$\rangle$
\end{itemize}

This is similar to the \texttt{indexonlyfirst} package option but only for entries that have a category with this attribute set to \texttt{true}.

\begin{itemize}
\item \texttt{wrgloss}=$\langle$value$\rangle$
\end{itemize}

When using the \texttt{\gls} or \texttt{\glstext}-like commands, this will automatically set \texttt{wrgloss=after} if this attribute is set to \texttt{“after”}.

\begin{itemize}
\item \texttt{textformat}=$\langle$cs-name$\rangle$
\end{itemize}

The \texttt{\gls} and \texttt{\glstext}-like commands have the link text encapsulated in the argument of \texttt{\glstextformat} by default (the outer formatting, see §5.5.1). If the \texttt{textformat} attribute is set, the control sequence given by the attribute value will be used instead. The attribute value should be the name (without the leading backslash) of a command that takes a single argument (the link text). Remember that the abbreviation styles may apply an additional font change.

\begin{itemize}
\item \texttt{hyperoutside}=$\langle$boolean$\rangle$
\end{itemize}

This boolean attribute may be \texttt{false}, \texttt{true} or unset. If unset, \texttt{true} is assumed. This indicates the default setting of the \texttt{hyperoutside} option, described in §5.1.

10.2.1.3. Glossary Attributes

\begin{itemize}
\item \texttt{glossdesc}=$\langle$value$\rangle$
\end{itemize}

This attribute is checked by the \texttt{\glossentrydesc} to determine whether or not to apply any case change. The value may be one of:

\texttt{firstuc}
Applies sentence case. That is, the first letter of the description will be converted to uppercase (using \Glsentrydesc).

**title**

Applies title case. If you have at least glossaries v4.48, the title casing is indirectly performed by \glscapitalisewords, which defaults to \capitalisewords (provided by mfirstuc). You can either redefine \glscapitalisewords if you want the change to also affect \glsentrytitlecase or if you only want the change to apply to the attribute case-changing then redefine \glsxtrfieldtitlecasecs. For example:

\[\newcommand*{\glsxtrfieldtitlecasecs}[1]{\xcapitalisefmtwords*{#1}}\]

(Note that the argument to \glsxtrfieldtitlecasecs will be a control sequence whose replacement text is the entry’s description, which is why \xcapitalisefmtwords is needed instead of \capitalisewords.)

If an error occurs with this setting, try redefining \glsxtrfieldtitlecasecs as shown above.

Any other values of this attribute are ignored. Remember that there are design limitations for both the sentence case and the title case commands. See the mfirstuc user manual for further details.

If you are using bib2gls, you can use the **description-case-change** setting instead.

\[\texttt{glossdescfont=\langle cs-name \rangle}\]

If set, the value should be the name of a control sequence (without the leading backslash) that takes one argument. This control sequence will be applied by \glossentrydesc to the description text. For example:

\[\texttt{\glssetcategoryattribute\{general\}\{glossdescfont\}\{emph\}}\]

\[\texttt{glossname=\langle value \rangle}\]
10. Categories

As `glossdesc` but applies to `\glossentryname`. Additionally, if this attribute is set to “uc” the name is converted to all caps.

If you are using `bib2gls`, you can use the `name-case-change` setting instead.

```
indexname={⟨value⟩}
```

If set, the post-name hook will index the entry using `\index`. See §12 for further details.

```
glossnamefont=⟨cs-name⟩
```

As `glossdescfont` but applies to `\glossentryname`. Note that this overrides `\glsnamefont` which will only be used if this attribute hasn’t been set.

Remember that glossary styles may additionally apply a font change, such as the list styles which put the name in the optional argument of `\item`.

```
glossymbolfont=⟨cs-name⟩
```

This is similar to `glossnamefont` and `glossdescfont` but is used by `\glossentrysymbol`.

10.2.1.4. Other Attributes

```
headuc=⟨boolean⟩
```

If this attribute is set to “true”, commands like `\glsfmtshort` will use the upper case version in the page headers.

```
entrycount=⟨trigger-value⟩
```

The value of this attribute (if set) must be an integer and is used in combination with `\glsenableentrycount` (see §6.1). Leave blank or undefined for categories that shouldn’t have this facility enabled. The value of this attribute is used by `\glsextrifcounttrigger` to determine how commands such as `\cgls` should behave.

```
linkcount=⟨boolean⟩
```

526
This attribute is set to true by \GlsXtrEnableLinkCounting (see §6.2).

\begin{verbatim}
linkcountmaster=(counter-name)
\end{verbatim}

This attribute is set by \GlsXtrEnableLinkCounting to the name of the counter that requires the link counter to be added to its reset list (see §6.2).

\begin{verbatim}
dualindex=(value)
\end{verbatim}

If this attribute is set, whenever a glossary entry has information written to the external glossary file through commands like \gls and \glsadd, a corresponding line will be written to the indexing file using \index. The value may be true to simply enable this feature or the value may be the encap to use with \index. See §12 for further details.

\begin{verbatim}
targeturl=(url)
\end{verbatim}

If set, the hyperlink generated by commands like \gls will be set to the URL provided by this attribute’s value. For example:

\begin{verbatim}
\glssetcategoryattribute{general}{targeturl}{master-doc.pdf}
\end{verbatim}

(See also the accompanying sample file sample-external.tex.) If the URL contains awkward characters (such as % or ~) remember that the base glossaries package provides commands like \glspercentchar and \glstildechar that expand to literal characters.

\begin{verbatim}
targetname=(anchor)
\end{verbatim}

If you want to a named anchor within the target URL (notionally adding #⟨name⟩ to the URL), then you also need to set targetname to the anchor ⟨name⟩. You may use \glslabel within ⟨name⟩ which is set by commands like \gls to the entry’s label.

All the predefined glossary styles start each entry listing with \glistarget which sets the anchor to \glolinkprefix\glslabel, so if you want entries to link to glossaries in the URL given by targeturl, you can just do:

\begin{verbatim}
\glssetcategoryattribute{general}{targetname}{\glolinkprefix\gls-
label}
\end{verbatim}

(If the target document changed \glolinkprefix then you will need to adjust the above as
appropriate.)

\texttt{targetcategory=⟨anchor⟩}

If the anchor is in the form \(⟨\text{name1}.\text{name2}⟩\) then use \texttt{targetname} for the \(⟨\text{name2}⟩\) part and \texttt{targetcategory} for the \(⟨\text{name1}⟩\) part.

For example:

\begin{verbatim}
\glssetcategoryattribute{general}{targeturl}{master-doc.pdf}
\glssetcategoryattribute{general}{targetcategory}{page}
\glssetcategoryattribute{general}{targetname}{7}
\end{verbatim}

will cause all link text for \texttt{general} entries to link to \texttt{master-doc.pdf#page.7} (page 7 of that PDF).

If you want a mixture in your document of entries that link to an internal glossary and entries that link to an external URL then you can use \texttt{\newignoredglossary*} for the external list. For example:

\begin{verbatim}
\newignoredglossary*{external}
\glssetcategoryattribute{external}{targeturl}{master-doc.pdf}
\glssetcategoryattribute{general}{targetname}{\glolinkprefix\glslabel}
\newglossaryentry{sample}{name={sample},description={local example}}
\newglossaryentry{sample2}{name={sample2},
type={external},
category={external},
description={external example}}
\end{verbatim}

\texttt{externallocation=⟨PDF filename⟩}

The value should be the file name of the target document when manually indexing an external location with \texttt{thevalue}. In general, it’s better to use \texttt{bib2gls v1.7+} which can handle multiple external sources and doesn’t require this attribute.
10. Categories

10.2.2. Accessing and Setting Attributes

Attributes can be set using the following commands:

\glssetcategoryattribute{\langle category\rangle}{\langle attribute\rangle}{\langle value\rangle}
Locally sets the given attribute to \langle value\rangle for the given category.

\glssetcategoriesattribute{\langle category list\rangle}{\langle attribute\rangle}{\langle value\rangle}
Globally sets the given attribute to \langle value\rangle for all the categories in the comma-separated list \langle category list\rangle.

\glssetcategoryattributes{\langle category\rangle}{\langle attribute list\rangle}{\langle value\rangle}
Globally sets each attribute in the comma separated \langle attribute list\rangle to \langle value\rangle for the given \langle category\rangle.

\glssetcategoriesattributes{\langle category list\rangle}{\langle attribute list\rangle}{\langle value\rangle}
Globally sets each attribute in the comma separated \langle attribute list\rangle to \langle value\rangle for each category in the comma-separated list \langle category list\rangle.

\glssetattribute{\langle entry-label\rangle}{\langle attribute\rangle}{\langle value\rangle}
Locally sets the given attribute to \langle value\rangle for the category associated with the entry identified by \langle entry-label\rangle. This command can’t be used to assign an attribute for a multi-entry category.

\glssetregularcategory{\langle category\rangle}
A shortcut that sets the regular attribute to true for the given category using \glssetcategoryattribute.

An attribute can be locally unset using:

\glsunsetcategoryattribute{\langle category\rangle}{\langle attribute\rangle}
10. Categories

Attribute values can be obtained with the following commands:

\glsgetcategoryattribute{⟨category⟩}{⟨attribute⟩}

Expands to the value of the given attribute for the given category. Expands to nothing if the attribute hasn’t been set.

\glsgetattribute{⟨entry-label⟩}{⟨attribute⟩}

Expands to the value of the given attribute for the category associated with the entry identified by ⟨entry-label⟩. Expands to nothing if the attribute hasn’t been set. This command can’t be used to assign an attribute for a multi-entry category.

Attributes can be tested with the following commands.

\glshascategoryattribute{⟨category⟩}{⟨attribute⟩}{⟨true⟩}{⟨false⟩}

This uses etoolbox’s \ifcsvoid and does ⟨true⟩ if the attribute has been set and isn’t blank and isn’t \relax otherwise it does ⟨false⟩.

\glshasattribute{⟨entry-label⟩}{⟨attribute⟩}{⟨true⟩}{⟨false⟩}

As \glshascategoryattribute but the category is obtained from the given entry. This command can’t be used to test an attribute associated with a multi-entry category.

\glsifcategoryattribute{⟨category⟩}{⟨attribute⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}

This tests if the given attribute for the given category is set and equal to ⟨value⟩. If true, ⟨true⟩ is done. If the attribute isn’t set or is set but isn’t equal to ⟨value⟩, ⟨false⟩ is done.

For example:

\glsifcategoryattribute{general}{nohyper}{true}{NO HYPER}{HYPER}

This does “NO HYPER” if the general category has the nohyper attribute set to true otherwise if does “HYPER”.

\glsifattribute{⟨entry-label⟩}{⟨attribute⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}
10. Categories

As \texttt{\textbackslash glsifcategoryattribute} but the category is obtained from the given entry. This command can’t be used to test an attribute associated with a multi-entry category.

\begin{verbatim}
\glsifregularcategory{⟨category⟩}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

A shortcut that tests if the given category has the regular attribute set to true.

\begin{verbatim}
\glsifnotregularcategory{⟨category⟩}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

A shortcut that tests if the given category has the regular attribute set to false.

\begin{verbatim}
\glsifregular{⟨entry-label⟩}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

As \texttt{\textbackslash glsifregularcategory} but the category is obtained from the given entry. This command can’t be used to test an attribute associated with a multi-entry category.

\begin{verbatim}
\glsifnotregular{⟨entry-label⟩}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

As \texttt{\textbackslash glsifnotregularcategory} but the category is obtained from the given entry. This command can’t be used to test an attribute associated with a multi-entry category.

\begin{verbatim}
\glsifcategoryattributetrue{⟨category⟩}{{⟨attribute⟩}}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

Expands to \texttt{⟨true⟩} if the attribute is true and \texttt{⟨false⟩} otherwise. Expands to \texttt{⟨false⟩} if there’s no such attribute for the given category.

\begin{verbatim}
\glsifattributetrue{⟨entry-label⟩}{{⟨attribute⟩}}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

As \texttt{\textbackslash glsifcategoryattributetrue} but the category is obtained from the given entry. Expands to \texttt{⟨false⟩} if the entry isn’t defined. This command can’t be used to test an attribute

\begin{verbatim}
\glsifregular{⟨category⟩}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

A shortcut that tests if the given category has the regular attribute set to true.  

\begin{verbatim}
\glsifnotregular{⟨category⟩}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

A shortcut that tests if the given category has the regular attribute set to false.

\begin{verbatim}
\glsifregular{⟨entry-label⟩}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

As \texttt{\textbackslash glsifregularcategory} but the category is obtained from the given entry. This command can’t be used to test an attribute associated with a multi-entry category.

\begin{verbatim}
\glsifnotregular{⟨entry-label⟩}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

As \texttt{\textbackslash glsifnotregularcategory} but the category is obtained from the given entry. This command can’t be used to test an attribute associated with a multi-entry category.

\begin{verbatim}
\glsifcategoryattributetrue{⟨category⟩}{{⟨attribute⟩}}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

Expands to \texttt{⟨true⟩} if the attribute is true and \texttt{⟨false⟩} otherwise. Expands to \texttt{⟨false⟩} if there’s no such attribute for the given category.

\begin{verbatim}
\glsifattributetrue{⟨entry-label⟩}{{⟨attribute⟩}}{{⟨true⟩}}{{⟨false⟩}}
\end{verbatim}

As \texttt{\textbackslash glsifcategoryattributetrue} but the category is obtained from the given entry. Expands to \texttt{⟨false⟩} if the entry isn’t defined. This command can’t be used to test an attribute
10. Categories

associated with a multi-entry category.

\glsifcategoryattributehasitem{\langle category \rangle}{\langle attribute \rangle}{\langle item \rangle}{\langle true \rangle}
{\langle false \rangle}

Does \langle true \rangle if the category has the attribute (whose value is a comma-separated list) contains the given item and \langle false \rangle otherwise. Does \langle false \rangle if there’s no such attribute for the given category. The item and list are expanded and passed to datatool’s \texttt{DTLifinlist} to perform the test.
11. **bib2gls: Managing Reference Databases**

The command line application `bib2gls` performs two functions in one:

- selects entries according to records found in the aux file (similar to `BibTeX`),
- hierarchically sorts entries and collates location lists (similar to `makeindex` or `xindy`).

Instead of storing all your entry definitions in a `tex` and loading them using `\input` or `\loadglsentries`, the entries can instead be stored in a `bib` file and `bib2gls` can selectively write the appropriate commands to a `glstex` file which is loaded using `\GlsXtr-LoadResources`.

This means that you can use a reference managing system to maintain the database and it reduces the `\LaTeX` overhead by only defining the entries that are actually required in the document. If you currently have a `tex` file that contains hundreds of definitions, but you only use a dozen or so in your document, then the build time is needlessly slowed by the unrequired definitions that occur when the file is input. (You can convert an existing `tex` file containing glossary definitions to a `bib` file using `convertgls2bib`, supplied with `bib2gls`.)

There are some new commands and options added to `glossaries-extra` to help assist the integration of `bib2gls` into the document build process.

This chapter just provides a general overview of `bib2gls`. The full details and some sample documents are provided in the `bib2gls` manual\footnote{mirrors.ctan.org/support/bib2gls/bib2gls.pdf}.

```tex
\texttt{texdoc bib2gls}
```

An example of the contents of `bib` file that stores glossary entries that can be extracted with `bib2gls` called, say, `terms.bib`:

```plaintext
@entry{bird,  
  name={bird},  
  description={feathered animal},  
  see={\[see also\]duck,goose}  
}
```

\footnote{mirrors.ctan.org/support/bib2gls/bib2gls.pdf}
@entry{duck, 
name={duck},
description={a waterbird with short legs} 
}

@entry{goose, 
name="goose",
plural="geese",
description={a waterbird with a long neck} 
}

The following provides some abbreviations in a file called, say, abbrvs.bib:

@string{ssi={server-side includes}}
@string{html={hypertext markup language}}

@abbreviation{shtml, 
short="shtml",
long= ssi # " enabled " # html,
description={a combination of \gls{html} and \gls{ssi}} 
}

@abbreviation{html, 
short ="html",
long = html,
description={a markup language for creating web pages} 
}

@abbreviation{ssi, 
short="ssi",
long = ssi,
description={a simple interpreted server-side scripting language} 
}

@abbreviation{xml, 
short={xml}, long={extensible markup language}, description={a simple text-base format for representing structured information} }

The above defines bib strings (with @string) and uses string concatenation (with #), which is a \LaTeX feature. Another supported bib feature is @preamble, which may be used to provide command definitions.

Here are some symbols in a file called, say, symbols.bib:
To ensure that bib2gls can find out which entries have been used in the document, you need the record package option:

```latex
\usepackage{record}{glossaries-extra}
```

If you are using hyperref, you may prefer to use record=nameref. The glstex file created by bib2gls is loaded using:

```latex
\glsxtrresourcefile[(options)]{(basename)}
```

where \texttt{(basename)} is the basename (without the extension) of the glstex file. This command will redefine \texttt{\glsindexingsetting} to \texttt{bib2gls} (or \texttt{bib2gls-xindy} or \texttt{bib2gls-makeindex} if record=hybrid). There's a shortcut version:

```latex
\GlsXtrLoadResources[(options)]
```

This internally uses \texttt{\glsxtrresourcefile} and sets the \texttt{(basename)} to \texttt{\jobname} in the first instance and to \texttt{\jobname-(n)} on subsequent instances (where \texttt{(n)} is incremented at the end of every \texttt{\GlsXtrLoadResources}). For example:
is equivalent to:

```
\usepackage[record]{glossaries-extra}
\GlsXtrLoadResources[src={terms,moreterms}]
\GlsXtrLoadResources[src={symbols,constants}]
\GlsXtrLoadResources[src={abbreviations}]
```

If required, the value of \( n \) is stored in the count register:

```
\glsxtrresourcecount
```

although there should be little need to use this.

Since \GlsXtrLoadResources is more convenient to use than \glsxtrresourcefile, all examples use \GlsXtrLoadResources.

The \glsxtrresourcefile command writes the following to the aux file:

```
\glsxtr@resource{⟨options⟩}{⟨basename⟩}
```

and will input \( ⟨filename⟩.glstex \) if it exists. (Version 1.08 assumed \( ⟨filename⟩.tex \) but that’s potentially dangerous if, for example, \( ⟨filename⟩ \) happens to be the same as \jobname. The glstex extension was enforced by version 1.11.)

If you are using or developing a build system that needs to know which applications to run as part of the document build, you can search the aux for for instances of \glsxtrresource. For example, using arara:

```
% arara: bib2gls if found("aux", "glsxtr@resource"
```

Since the glstex file won’t exist on the first \LaTeX run, the record package option additionally switches on undefaction=warn. Any use of commands like \gls or \glstext will produce ?? in the document, since the entries are undefined at this point. Once bib2gls has created the glstex file the references should be resolved. This may cause a shift in the
locations if the actual text produced once the entry is defined is significantly larger than the placeholder ?? (as this can alter the page breaking).

Note that as from v1.12, \glsxtrresourcefile temporarily switches the category code of @ to 11 (letter) while it reads the file to allow for any internal commands.

The package options record=only and record=nameref automatically load glossaries \-extra\-bib2gls, which provides additional commands that are useful with bib2gls. Since they cover sorting and location lists, they’re not relevant with the record=hybrid option.

These commands are provided by glossaries-extra for use with bib2gls. The information provided with \GlsXtrLoadResources is written to the aux file using:

\protected@write\@auxout{\glsxtrresourceinit}{⟨information⟩}

where ⟨information⟩ is the information to pass to bib2gls. The command in the second argument:

\glsxtrresourceinit

may be used to temporarily redefine commands before the information is written to the file. This does nothing by default, but may be redefined to allow the use of short commands for convenience. For example, with:

\renewcommand{\glsxtrresourceinit}{\let\u\glshex}

you can just use, for example, \u E6 instead of \string\uE6 in the custom rule. This redefinition of \u is scoped so its original definition is restored after the write operation.

If you have complex regular expressions or use assign-fields (bib2gls v3.3+), you may find it more convenient to redefine \glsxtrresourceinit to use \GlsXtrResourceInit-EscSequences.

\glsxtrMFUsave

If you have mfirstuc v2.08+, this command will be used on the first instance of \glsxtrresourcefile, and will add \MFUsave to the begin document hook and then disable itself. This is provided to help bib2gls v3.0+ pick up any of mfirstuc’s exclusions, blockers and mappings to assist with its sentence case function. The assumption is that all exclusions, blockers and mappings will be set up in the preamble. If there are any within the document environment that you want bib2gls to be aware of, redefine this command to do nothing.
before the first instance of \glsxtrresourcefile (or \GlsXtrLoadResources) and use \MFUsaveatend instead.

If you have multiple resource commands and you want a default set of options you can supply them in the definition of:

\GlsXtrDefaultResourceOptions

For example:

\renewcommand{\GlsXtrDefaultResourceOptions}{selection=all}

This should be done before the resource commands to which the options should apply.

### 11.1. Indexing (Recording)

As with makeindex and xindy, the \gls-like and \glstext-like commands automatically index, but the underlying indexing mechanism is more like that used with \makenoidx-glossaries. Each indexing instance creates a record in the aux file, which bib2gls can then pick up when it parses the aux file. Each record has an associated format (the location encap) which can be set with the format key and an associated location counter (as with the other indexing methods).

The formatted location list is stored in the location field (unless save-locations=false). Additionally, the individual locations are stored in the loclist field as an etoolbox internal list (as with \makenoidxglossaries). This may be used to pick out individual locations to avoid the complexity of parsing the formatted list.

See the bib2gls manual for information on how to separate the location list into groups associated with different counters.

### 11.2. Selection

The default behaviour is for bib2gls to select all entries that have a record in the aux file, and any dependent entries (including parent and cross-references). The glsignore format (for example, \gls[format=glsignore]{duck}) is recognised by bib2gls as a special ignored record. This means that it will match the selection criteria but the record won’t be added to the location list. This means that you won’t get spurious commas in the location list (as can happen with the other indexing methods), so you can do, for example,

\GlsXtrSetDefaultNumberFormat{glsignore}

at the start of the front matter and
11. bib2gls: Managing Reference Databases

\GlsXtrSetDefaultNumberFormat{glsnumberformat}

at the start of the main matter to prevent any records in the front matter from occurring in
the location lists.

Commands like \glsaddall and \glsaddallunused don’t work with bib2gls as
the command has to iterate over each glossary’s internal lists of defined entry labels,
which will be empty on the first run and on subsequent runs will only contain those
entries that have been selected by bib2gls. Use selection=all to select all entries
instead.

The selection option indicates which entries should be selected from the bib files (listed
in src). For example, selection=all indicates to select all entries, regardless of whether
or not the entries have been referenced in the document. This will lead to empty location
lists for some (or all) entries. The default setting is selection=recorded and deps, which
indicates to select all entries that have records and any dependent entries. See the bib2gls
user manual for more details of this option.

11.3. Sorting and Displaying the Glossary

With makeindex and xindy, the terms (read from the associated input file) are sorted and
the code to typeset the glossary is written to an output file, which is then input by \print-
glossary. With bib2gls, the entries supplied in the bib files are sorted and the entry
definition code (\longnewglossaryentry or \newabbreviation) is written to the glstex
file in the order obtained by sorting. This means that the glossary’s internal list is in the
required order, so the glossary can be displayed with \printunsrtglossary (see §8.4).

The indexing information, such as the location list or letter groups, is stored in fields such
as location or group (where applicable), so the information can be included by \print-
unsrtglossary, but it means that the information is also available for use elsewhere in the
document (so the sav_enumberlist package option provided by glossaries is redundant).

There are many sorting options provided by bib2gls. The default is to sort according to
the system locale. If the document has a language setting, you can use sort=doc to instruct
bib2gls to sort according to that. (The language tag obtained from tracklang’s interface is
written to the aux file.) For a multilingual document you need to explicitly set the locale
using a well-formed language tag. For example:

\GlsXtrLoadResources[
  src=terms, % data in terms.bib
  sort=de-DE-1996 % sort according to this locale
]
11. bib2gls: Managing Reference Databases

The locale-sensitive sort methods usually ignore most punctuation so for lists of symbols you may find it more appropriate to use one of the letter-base sort methods that sort according to the Unicode value of each character. Alternatively you can provide a custom rule. See the bib2gls manual for full details of all the available sort methods.

Suppose the bib examples shown earlier have been stored in the files terms.bib, abbrvs.bib and symbols.bib which may either be in the current directory or on \TeX’s path. Then the document might look like:

\documentclass{article}
\usepackage[record]{glossaries-extra}
\setabbreviationstyle{long-short-sc-desc}
\GlsXtrLoadResources[src=terms,abbrvs,symbols]
\begin{document}
\gls{bird}
\gls{shtml}
\gls{M}
\gls{printunsrtglossaries}
\end{document}

The document build process (assuming the document is called mydoc) is:

pdflatex mydoc
bib2gls mydoc
pdflatex mydoc
Example 151: Using *bib2gls*: one resource set

```
bird
server-side includes enabled hypertext markup language (SHTML)
M
```

**Glossary**

*bird*  feathered animal 1, *see also* duck & goose

duck  a waterbird with short legs

goose  a waterbird with a long neck

**hypertext markup language (html)**  a markup language for creating web pages 1

*M*  a matrix 1

**server-side includes enabled hypertext markup language (shtml)**  a combination of hypertext markup language (HTML) and server-side includes (ssi) 1

**server-side includes (ssi)**  a simple interpreted server-side scripting language 1

This creates a single glossary containing the entries: *bird*, *duck*, *goose*, *html*, *M*, *shtml* and *ssi* (in that order). The *bird*, *shtml* and *M* entries were added because *bib2gls* detected (from the aux file) that they had been used in the document. The other entries were added because *bib2gls* detected (from the *bib* files) that they are referenced by the used entries. In the case of *duck* and *goose*, they are in the *see* field for *bird*. In the case of *ssi* and *html*, they are referenced in the *description* field of *shtml*. These cross-referenced entries won’t have a location list when the glossary is first displayed, but depending on how they are referenced, they may pick up a location list on the next document build. The *xml* entry isn’t required at all, and so hasn’t been defined (from LaTeX’s point of view).

The entries can be separated into different glossaries with different sort methods:

```
\usepackage[record,abbreviations,symbols]{glossaries-extra}
\setabbreviationstyle{long-short-sc-desc}
\GlsXtrLoadResources[src=terms,sort=en-GB,type=main]
```
\begin{document}
\texttt{GlsXtrLoadResources}
\texttt{[src=abbrvs,sort=letter-nocase,type=abbreviations]}

\texttt{GlsXtrLoadResources}
\texttt{[src=symbols,sort=use,type=symbols]}

\texttt{\begin{document}}
\texttt{\gls{bird}}

\texttt{\gls{shtml}}

\texttt{\gls{M}}

\texttt{\printunsrtglossaries}
\texttt{\end{document}}
Example 152: Using \bib2gls: multiple resource sets

\begin{verbatim}
  bird
  server-side includes enabled hypertext markup language (shtml)
  \end{verbatim}

\section*{Glossary}

\textbf{bird} feathered animal 1, see also duck \& goose

\textbf{duck} a waterbird with short legs

\textbf{goose} a waterbird with a long neck

\section*{Symbols}

\textbf{M} a matrix 1

\section*{Abbreviations}

\textbf{hypertext markup language (html)} a markup language for creating web pages 1

\textbf{server-side includes enabled hypertext markup language (shtml)} a combination of hypertext markup language (HTML) and server-side includes (SSI) 1

\textbf{server-side includes (ssi)} a simple interpreted server-side scripting language 1

Or you can have multiple instance of \texttt{\GlsXtrLoadResources} with the same \texttt{type}, which will produce a glossary with ordered sub-blocks. For example:

\begin{verbatim}
\usepackage[record,style=indexgroup]{glossaries-extra}
\setabbreviationstyle{long-short-sc-desc}
\GlsXtrLoadResources
  [src=abbrvs,sort=letter-nocase,type=main,group=abbreviations]
\end{verbatim}
This sets the `group` field for each resource set to the label given by the `group` resource option.

This will result in a glossary where the first group has the label `abbreviations` and title “Abbreviations”, the second group has the label `symbols` and title “Symbols” and then follow the usual letter groups. Note that for this example to work, you must run `bib2gls` with the --group (or -g) switch. For example, if the document is called `myDoc.tex`:

```
pdflatex myDoc
bib2gls -g myDoc
pdflatex myDoc
```
Example 153: Using \texttt{bib2gls}: sub-blocks

\begin{verbatim}
  bird
  server-side includes enabled hypertext markup language (shtml)
  \textit{M}
\end{verbatim}

\textbf{Glossary}

\textbf{Abbreviations}

\textbf{hypertext markup language (html)} a markup language for creating web pages \textit{1}

\textbf{server-side includes enabled hypertext markup language (shtml)} a combination of hypertext markup language (HTML) and server-side includes (SSI) \textit{1}

\textbf{server-side includes (ssi)} a simple interpreted server-side scripting language \textit{1}

\textbf{Abbreviations}

\textbf{M} a matrix \textit{1}

\textbf{B}

\textbf{bird} feathered animal \textit{1}, \textit{see also} duck & goose

\textbf{D}

\textbf{duck} a waterbird with short legs

\textbf{G}

\textbf{goose} a waterbird with a long neck

The value of the \texttt{group} field must always be a label (so avoid special characters). You can set the corresponding title with \texttt{glsxtrsetgrouptitle} (see §8.6). If no title is set then the label is used as the group title.

You can provide your own custom sort rule. For example, if you are using \texttt{Xe\LaTeX} or \texttt{Lua\LaTeX}:

\texttt{545}
With old versions of the \LaTeX kernel, UTF-8 characters, such as é or ø, will expand when written to the aux file.

Some of the options, including `sort-rule`, allow Unicode characters to be indicated in the format `\u{hex}` (or `\u{hex}`). `bib2gls` will recognise this as the character given by the hexadecimal value `{hex}`.

The `options` provided to \GlsXtrLoadResources will expand as they are written to the aux file (unless protected). This includes `\u`, so with a non-Unicode aware engine or where the document source is required to be ASCII, the character æ needs to be written as `\string\u E6` and so on. Alternatively, use the shortcut `\string\u E6`.

For example, the above can be rewritten as:

```
\GlsXtrLoadResources[
  src={terms}, % entries in terms.bib
  sort=custom, % custom sort rule
  sort-rule=% required with sort=custom
  < æ;Æ < a;ã;ä;Å;Å;Ä;Ä < b,B
  < c;ć,C;Ć < d,D < e;é,E;É < f,F < g,G
  < h,H < i;í,I;Í < j,J < l,Ł < m,M < n,N
  < o;ö;ø,O;Ö;Ø < p,P < q,Q < r,R < s;ś,S;Ś < t,T < u;ú,U;Ú < v,V < w,W < x,X < y,Y < z;ż,Z;Ż
}\]
```
11. **bib2gls: Managing Reference Databases**

11.4. **Record Counting**

As from version 1.1 of bib2gls, you can save the total record count for each entry by invoking bib2gls with the --record-count or --record-count-unit switches. These options will ensure that when each entry is written to the glstex file bib2gls will additionally set the following internal fields for that entry:

- **recordcount**: set to the total number of records found for the entry;
- **recordcount⟨counter⟩**: set to the total number of records found for the entry for the given counter.

If --record-count-unit is used then additionally:

- **recordcount⟨counter⟩⟨location⟩**: set to the total number of records found for the entry for the given counter with the given location.

Only use the unit counting option if the locations don’t contain any special characters. With hyperref use \(\text{the}⟨\text{counter-name}⟩\) rather than \(\text{the}⟨\text{counter-name}⟩\). Otherwise, if you really need unit counting with locations that may contain formatting commands, then you can try redefining:

\[
\text{glsxtrdetoklocation}{⟨\text{location}⟩}
\]

so that it detokenizes ⟨location⟩ but take care when using \GlsXtrLocationRecordCount with commands like \the\text{the}⟨\text{counter-name}⟩ as they can end up becoming detokenized too early.

Note that the record count includes locations that bib2gls discards, such as ignored records, duplicates and partial duplicates (unless you filter them out with --record-count-rule). It doesn’t include cross-reference records. For example, suppose a document has an entry with the label bird that is recorded (indexed) as follows:

**Page 1** two (2) instances of \gls{bird};

**Page 2** one (1) instance of \gls{bird};

**Page 3** four (4) instances of \gls{bird};
Section 3  one (1) instance of \gls{counter=section}{bird}.

Then the total record count (stored in the \texttt{recordcount} field) is $2 + 1 + 4 + 1 = 8$, the total for the page counter (stored in the \texttt{recordcount.page} field) is $2 + 1 + 4 = 7$, and the total for the section counter (stored in the \texttt{recordcount.section} field) is 1.

With the unit counting on as well, the following fields are assigned:

- \texttt{recordcount.page.1} is set to 2;
- \texttt{recordcount.page.2} is set to 1;
- \texttt{recordcount.page.3} is set to 4;
- \texttt{recordcount.section.3} is set to 1.

You can access these fields using the following commands which will expand to the field value if set or to 0 if unset:

\begin{itemize}
  \item \texttt{\GlsXtrTotalRecordCount{⟨entry-label⟩}}
  \item \texttt{\GlsXtrRecordCount{⟨entry-label⟩}{⟨counter⟩}}
  \item \texttt{\GlsXtrLocationRecordCount{⟨entry-label⟩}{⟨counter⟩}{⟨location⟩}}
\end{itemize}

This expands to the total record count for the entry given by \texttt{⟨label⟩}. For example:

\begin{itemize}
  \item \texttt{\GlsXtrTotalRecordCount{bird}}
  \item \texttt{\GlsXtrRecordCount{bird}{page}}
  \item \texttt{\GlsXtrRecordCount{bird}{section}}
\end{itemize}

expands to 8.

expands to 7 and

expands to 1.
This expands to the total for the given location. For example

\GlsXtrLocationRecordCount{bird}{page}{3}

expands to 4. Be careful about using \thepage in the \langle location \rangle part. Remember that due to \TeX's asynchronous output routine, \thepage may not be correct.

There are commands analogous to the entry counting commands like \cgls and \cgls-format that are triggered by the record count. These are listed below. The test to determine if the entry's record count exceeds the trigger value (which should be stored in the record-count attribute) is obtained with:

\glsxtrifrecordtrigger{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}

If the recordcount attribute is set and ⟨total⟩ exceeds the value given by the recordcount attribute, then this does ⟨true⟩ otherwise it does ⟨false⟩. The ⟨total⟩ is given by:

\glsxtrrecordtriggervalue{⟨entry-label⟩}

This should expand to the record count value that needs testing. The default definition is:

\newcommand*{\glsxtrrecordtriggervalue}[1]{{\GlsXtrTotalRecordCount{#1}}}

This command may be redefined as appropriate. For example, it may be redefined to use \GlsXtrRecordCount for a particular location counter or to use \GlsXtrLocationRecordCount for a particular location.

The recordcount attribute may be set with \glssetcategoryattribute or can be set for each listed category with:

\GlsXtrSetRecordCountAttribute{⟨category-list⟩}{⟨value⟩}

The value must be an integer.

Commands like \rgls behave slightly differently to \cgls. It's necessary for the command to add a record to the aux file in order for the entry to be selected and for the record count to be correct on the next \bib2gls+\TeX run (for the default selection={recorded and deps}). The trigger record is created with format=glstriggerrecordformat, which \bib2gls v1.1+ recognises as a special type of ignored location format. This corresponds to
the command:

\glstriggerrecordformat{\textit{location}}

As with \glsignore, this command does nothing and is considered an ignored record (so it won't appear in the location list), but it indicates to \bib2gls that the entry must be selected and, if the trigger-type option has been set, the entry will be assigned to the trigger-type glossary.

For example, to assign the entry to an ignored glossary:

\newignoredglossary{ignored}
\GlsXtrLoadResources[trigger-type=ignored]

This ensures that the entry is defined but it won’t show up the normal glossary.

The post-link hook won’t be implemented if the record trigger is tripped. (That is, if the \rglsformat-like command is used instead of the \gls-like command.)

\rgls[\textit{options}]{\textit{entry-label}}[\textit{insert}]

modifiers: * + \textit{alt-mod}

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \gls otherwise it creates a trigger record and uses:

\rglsformat{\textit{entry-label}}{\textit{insert}}

This has the same definition as \cglssformat.

\rglsp{\textit{options}]{\textit{entry-label}}[\textit{insert}]

modifiers: * + \textit{alt-mod}

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \glspl otherwise it creates a trigger record and uses:

\rglspformat{\textit{entry-label}}{\textit{insert}}
11. bib2gls: Managing Reference Databases

which uses the appropriate plural fields.

\begin{align*}
\texttt{\textbackslash Gls}[\langle \text{options} \rangle]{\langle \text{entry-label} \rangle}\{\langle \text{insert} \rangle\} \\
\text{modifiers: } * + \langle \text{alt-mod} \rangle
\end{align*}

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \texttt{\Gls} otherwise it creates a trigger record and uses:

\begin{align*}
\texttt{\textbackslash Glsformat}\{\langle \text{entry-label} \rangle\}{\langle \text{insert} \rangle}\}
\end{align*}

which performs the appropriate case-change.

\begin{align*}
\texttt{\textbackslash Glsp1}[\langle \text{options} \rangle]{\langle \text{entry-label} \rangle}\{\langle \text{insert} \rangle\} \\
\text{modifiers: } * + \langle \text{alt-mod} \rangle
\end{align*}

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \texttt{\Glsp1} otherwise it creates a trigger record and uses:

\begin{align*}
\texttt{\textbackslash Glsp1format}\{\langle \text{entry-label} \rangle\}{\langle \text{insert} \rangle}\}
\end{align*}

which uses the appropriate plural fields and case-change.

\begin{align*}
\texttt{\textbackslash GLS}[\langle \text{options} \rangle]{\langle \text{entry-label} \rangle}\{\langle \text{insert} \rangle\} \\
\text{modifiers: } * + \langle \text{alt-mod} \rangle
\end{align*}

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \texttt{\GLS} otherwise it creates a trigger record and uses:

\begin{align*}
\texttt{\textbackslash GLSformat}\{\langle \text{entry-label} \rangle\}{\langle \text{insert} \rangle}\}
\end{align*}

which performs the appropriate case-change.

\begin{align*}
\texttt{\textbackslash GLSp1}[\langle \text{options} \rangle]{\langle \text{entry-label} \rangle}\{\langle \text{insert} \rangle\} \\
\text{modifiers: } * + \langle \text{alt-mod} \rangle
\end{align*}

If the value has been supplied by the recordcount attribute and is exceeded, this behaves like \texttt{\GLSp1} otherwise it creates a trigger record and uses:

\begin{align*}
\texttt{\textbackslash GLSp1format}\{\langle \text{entry-label} \rangle\}{\langle \text{insert} \rangle}\}
\end{align*}
which uses the appropriate plural fields and case-change.

To make it easier to switch on record counting for an existing document, you can use:

\glsxtrenablerecordcount

This redefines \gls, \glspl, \Gls, \GLS, \gls, \Glspl to \rgls, \rglsp, \rgls, \Glspl, \rGLS, \rGLSp1, respectively, for convenience. This command will also switch the shortcut commands such as \ac or \ab, if they have been enabled, from using the \cglslike commands to the corresponding \rgls command.

For example, using the earlier terms.bib, abbrvs.bib and symbols.bib example files:

\documentclass{article}
\usepackage[colorlinks]{hyperref}
\usepackage[record]{glossaries-extra}
\newignoredglossary{ignored}
\setabbreviationstyle{long-short-sc-desc}
\GlsXtrLoadResources[src={terms,abbrvs,symbols},
    trigger-type=ignored,
    category={same as entry}]
\glsxtrenablerecordcount
\GlsXtrSetRecordCountAttribute{general,abbreviation}{1}
\glsdefpostlink{entry}{\glsxtrpostlinkAddDescOnFirstUse}
\glsdefpostlink{symbol}{\glsxtrpostlinkAddDescOnFirstUse}
\begin{document}
\gls{bird}, \gls{ssi}, \gls{bird}, \gls{html}, \gls{M}, \gls{html}.
\printunsrtglossaries
\end{document}

If the document is called myDoc.tex, then the build process is:
The category={same as entry} resource option assigns the category field to the bib entry type (without the initial @). This means that the entries defined in terms.bib (with @entry) have their category set to entry, the entries defined in abbrvs.bib (with @abbreviation) have their category set to abbreviation, and the entries defined in symbols.bib (with @symbol) have their category set to symbol.

I’ve added post-link hooks to the entry and symbol categories to show the description on first use (but not for the abbreviation category).

Example 154: Using bib2gls: record counting

\begin{itemize}
\item \textit{M} (a matrix), server-side includes, \textit{bird} (feathered animal), \textit{hypertext markup language (HTML)}, \textit{M}, HTML.
\end{itemize}

**Glossary**

- **bird** feathered animal 1, see also duck & goose
- **duck** a waterbird with short legs
- **goose** a waterbird with a long neck
- **hypertext markup language (html)** a markup language for creating web pages 1
- **M** a matrix 1

In the above, ssi and bird only have one record. However, they have been treated differently. The ssi entry is using \texttt{\textbackslash{}glstformat} whereas the bird entry is using the normal \texttt{\textbackslash{}gls} behaviour. This is because the record counting hasn’t been applied to the custom entry category, whereas it has been applied to the abbreviation and symbol categories.

### 11.4.1. Unit Record Counting

If you want unit record counting you need to remember to invoke bib2gls with --record-count-unit and you will also need to redefine \texttt{\textbackslash{}glsxtrrecordtriggervalue} appropriately. For example, suppose you want to reset all abbreviations at the start of each chapter, so that the full form is shown again, but only if the abbreviation isn’t used elsewhere in the chapter.
This would require records with the `counter` set to chapter. This can be done with the `counter` package option:

\usepackage[record,counter=chapter]{glossaries-extra}

However, this will have chapter numbers instead of page numbers in the location lists. If you don’t want location lists then this isn’t a problem. The list can simply be suppressed with `nonumberlist`.

If you want page numbers in the location lists then you will need to record each entry with both the page and chapter counters. This can be done with the hook that occurs before the \gls options are set:

\renewcommand{\glslinkpresetkeys}{\%
    \glsadd[format=glsignore,counter=chapter]{\glslabel}}

Note that I’ve used the ignored location format to prevent the chapter number from being added to the location list. An alternative is to use the `loc-counters=page` resource option to only show the locations that use the page counter.

The definition of `\glsxtrrecordtriggervalue` needs to be changed so that it uses the total for the given location. If `hyperref` is used, you will need `\theHchapter`:

\renewcommand*{\glsxtrrecordtriggervalue}[1]{\%
    \GlsXtrLocationRecordCount{#1}{chapter}{\theHchapter}%
    }

otherwise use `\thechapter`.

Consider the following (using the abbreviations defined in the earlier `abbrvs.bib`):

\begin{document}
\chapter{First}
\gls{html}. \gls{html}. \gls{html}. \gls{ssi}.

\chapter{Second}
\gls{html}. \gls{ssi}. \gls{ssi}.
\gls{xml}.

\printunsrtglossaries
\end{document}

Note that the `xml` entry is only used once in the entire document, but it will still be added to
the glossary.

The previous example used the \texttt{trigger-type} resource option to move entries with the \texttt{glstriggerrecordformat} \texttt{encap} (that is, they didn’t exceed the trigger value) to another glossary. Unfortunately, using that option in this case will move all three abbreviations to the \texttt{trigger-type} glossary. The \texttt{ssi} entry is only used once in the first chapter (but is used twice in the second chapter), and the \texttt{html} is only used once in the second chapter (but is used three times in the first chapter). So all three will have records in the aux file with the special \texttt{glstriggerrecordformat} format.

A simple solution is to omit any entries that don’t have the \texttt{location} field set when displaying the glossary:

\begin{verbatim}
\renewcommand*{\printunsrtglossaryentryprocesshook}[1]{%
  \glsxtrifhasfield*{location}{#1}{
    {\printunsrtglossaryskipentry}\%
  }
\printunsrtglossaries
\end{verbatim}

An alternative is to test the total record count, but remember that each entry is being recorded twice: once with the page counter and once with the chapter counter, so the total count for the \texttt{ssi} entry will be 2 not 1.

\begin{verbatim}
\end{verbatim}

Take care not to strip entries from a hierarchical glossary as it will break the hierarchy and will cause formatting problems in the glossary.

The complete document is:

\begin{verbatim}
\documentclass{scrreport}
\usepackage[T1]{fontenc}
\usepackage[colorlinks]{hyperref}
\usepackage[record,postdot]{glossaries-extra}
\setabbreviationstyle{long-short-sc-desc}
\GlsXtrLoadResources[src={abbrvs}]
\preto{\chapter}{\glsresetall}
\glsxтренablerecordcount
\GlsXtrSetRecordCountAttribute{abbreviation}{1}
\renewcommand{\glslinkpresetkeys}{%}
\end{verbatim}
11. bib2gls: Managing Reference Databases

\begin{document}
\chapter{First}
\gls{html}. \gls{html}. \gls{html}. \gls{ssi}.

\chapter{Second}
\gls{html}. \gls{ssi}. \gls{ssi}. \gls{xml}.

\renewcommand*{\printunsrtglossaryentryprocesshook}[1]{%
  \glsxtrifhasfield*{location}{#1}
  {}{\printunsrtglossaryskipentry}
}
\printunsrtglossaries
\end{document}

If the document is in a file called myDoc.tex then the document build is:

\texttt{pdflatex myDoc}
\texttt{bib2gls --record-count-unit myDoc}
\texttt{pdflatex myDoc}

Example 155: Using bib2gls: unit record counting

<table>
<thead>
<tr>
<th></th>
<th>First</th>
<th>Second</th>
<th>Glossary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>html</td>
<td>ssi</td>
<td>html</td>
</tr>
<tr>
<td>2</td>
<td>html</td>
<td>xml</td>
<td>ssi</td>
</tr>
</tbody>
</table>

11.4.2. Mini-Glossaries

Record counting doesn’t have to be used with the \rGLS set of commands. When bib2gls writes the code to the glstex file to save the record counting information, it does it with helper commands that it provides in the glstex file:

\bibglsettotalrecordcount{⟨entry-label⟩}{⟨value⟩}

This sets the total record count and is defined in the glstex file as:
This sets the total for the given counter and is defined as:

\providecommand*{\bibglssetrecordcount}{2}{\GlsXtrSetField{#1}{recordcount}.{#2}}%

The following is only available with \texttt{--record-count-unit}:

\bibglssetlocationrecordcount{⟨entry-label⟩}{⟨counter⟩}{⟨location⟩}{⟨value⟩}

This sets the total for the given location and is defined as:

\providecommand*{\bibglssetlocationrecordcount}{4}{\GlsXtrSetField{#1}{recordcount.⟨#2⟩.⟨#3⟩}}%
\provideignoredglossary{minigloss.⟨#2⟩.⟨#3⟩}
\glsxtrcopytoglossary{#1}{minigloss.⟨#2⟩.⟨#3⟩}

By defining one of more of these commands before the \texttt{glstex} file is input, it’s possible to pick up the information, without the need to iterate over all entries later. For example, the following will create a mini-glossary for each particular location and populate it with entries that have a record for that location.

\newcommand*{\bibglssetlocationrecordcount}{4}{\GlsXtrSetField{#1}{recordcount.⟨#2⟩.⟨#3⟩}}%
\provideignoredglossary{minigloss.⟨#2⟩.⟨#3⟩}
\glsxtrcopytoglossary{#1}{minigloss.⟨#2⟩.⟨#3⟩}

I’ve omitted \texttt{\glsxtrdetoklocation} for clarity and because I’m confident the locations won’t be problematic. The mini-glossary can then be displayed at the start of the chapter with:
The previous example can be altered to strip the \rgls commands and instead add a mini-glossary at the start of each chapter (the redefinition of \glslinkpresetkeys remains to ensure there are locations with the chapter counter). I’ve also provided a command to make it easier to display the mini-glossaries:

\newcommand{\minigloss}{%}
\printunsrtglossary*[style=abbr-short-long,type=minigloss.chapter.\theHchapter,groups=false,target=false]{{\renewcommand{\glossarysection}[2][]{}\renewcommand{\glslongextraShortLongTabularHeader}{\toprule}}%}

The document build is the same.

Example 156: Using bib2gls: unit record counting mini-glossary

<table>
<thead>
<tr>
<th>1 First</th>
<th>2 Second</th>
<th>Glossary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssi server-side includes</td>
<td>ssi server-side includes (ssi)</td>
<td>ssi server-side includes</td>
</tr>
<tr>
<td>xml extensible markup language</td>
<td>xml extensible markup language (xml)</td>
<td>xml extensible markup language</td>
</tr>
</tbody>
</table>

11.5. The glossaries-extra-bib2gls package

\usepackage[glossaries-extra-bib2gls]
\usepackage[record]{glossaries-extra} or \usepackage[record=nameref]{glossaries-extra}

The package options record=only (or simply record) and record=nameref automatically loads the supplementary package glossaries-extra-bib2gls, which provides some commands that are specific to bib2gls, in particular to sorting and location lists which aren’t relevant with record=hybrid.

If glossaries-extra-bib2gls is loaded via the record package option then the check for associated language files (see §15) will also search for the existence of glossaries-extr-⟨script⟩.ldf for each document dialect (where ⟨script⟩ is the four letter script identifier, such as Latn).
11.5.1. Displaying Glossaries

Glossaries are displayed with the "unsrt" family of commands (see §8.4). Some styles, such as bookindex, are customized for use with \bib2gls.

The following commands are shortcuts that use \printunsrglossary. However, they are only defined if a corresponding package option has been set before \glossaries-extra-\bib2gls is loaded. This means that the options must be passed as a package option, not using \glossariesextrasetup, if the shortcut commands are required.

\begin{itemize}
  \item \texttt{\printunsrtabbreviations[⟨options⟩]}
  \item This shortcut is provided if the \texttt{abbreviations} package option has been used. This is a shortcut for:
    \begin{itemize}
      \item \texttt{\printunsrglossary[\texttt{type=abbreviations,⟨options⟩}]}
    \end{itemize}
  \item \texttt{\printunsrtacronyms[⟨options⟩]}
  \item This shortcut is provided if the \texttt{acronyms} (or \texttt{acronym}) package option has been used. This is a shortcut for:
    \begin{itemize}
      \item \texttt{\printunsrglossary[\texttt{type=\texttt{acronym}type,⟨options⟩}]}
    \end{itemize}
  \item \texttt{\printunsrtsymbols[⟨options⟩]}
  \item This shortcut is provided if the \texttt{symbols} package option has been used. This is a shortcut for:
    \begin{itemize}
      \item \texttt{\printunsrglossary[\texttt{type=symbols,⟨options⟩}]}
    \end{itemize}
  \item \texttt{\printunsrtnumbers[⟨options⟩]}
  \item This shortcut is provided if the \texttt{numbers} package option has been used. This is a shortcut for:
\end{itemize}
11. `bib2gls`: Managing Reference Databases

\[\text{\texttt{\textbackslash printunsrtglossary[type=numbers,\{options\}]}}\]

\[\text{\texttt{\textbackslash printunsrtindex[\{options\}]}}\]

This shortcut is provided if the `index` package option has been used. This is a shortcut for:

\[\text{\texttt{\textbackslash printunsrtglossary[type=index,\{options\}]}}\]

11.5.2. Helper Commands for Resource Options

\[\text{\texttt{\textbackslash glshex\{hex\}}}\]

This simply expands to `\texttt{\textbackslash string\texttt{\textbackslash u\{hex\}}}`, which is used to identify the Unicode character `\{hex\}` in the value of some resource options.

\[\text{\texttt{\textbackslash glhashchar}}\]

This expands to a literal `#` character (similar to `\texttt{\textbackslash glsbackslash}`).

\[\text{\texttt{\textbackslash glscapturedgroup\{n\}}}\]

This simply expands to `\texttt{\textbackslash string\texttt{\textbackslash$\{n\}}}` which is used to indicate the \{n\}th captured group in a regular expression replacement in the value of some resource options (requires `bib2gls` v1.5+), such as `sort-replace`. For example:

\[\text{\texttt{sort-replace=\{([a-zA-Z])\texttt{\textbackslash string\texttt{\textbackslash.}}\{\texttt{\textbackslash glscapturedgroup1}\}}}\]

This removes a full stop that follows any of the characters a,...,z or A,...,Z.

Note that `\texttt{\textbackslash glscapturedgroup}` isn’t the same as the match group identifier `\texttt{\textbackslash MGP}`.

If you have complex regular expressions or use `assign-fields`, you may find it more...
convenient to redefine \glsxtrresourceinit to use the following command:

\GlsXtrResourceInitEscSequences

\GlsXtrResourceInitEscSequences should not be used outside of the definition of \glsxtrresourceinit as the definitions will likely cause interference and are only intended as resource option instructions for bib2gls.

This command locally redefines escape sequences used in regular expressions so that they detokenize when they expand. This means that you won’t need to use \string or \protect in front of them. The following commands are defined:

- \u⟨hex⟩ (Unicode character);
- General use: \cs{⟨csname⟩} (expands to detokenized ⟨csname⟩ when writing to the aux file);
- For literal characters in regular expressions: \ . \ / \ | \ & \ + \ < \ > \ * \ $ \ ^ \ ~ \ ( \ ) \ [ \ ] \ " \ - \ ? \ : \ #
- Special markup for use as instructions or keywords in assign-fields: \MGP \LEN \CAT \TRIM \CS \INTERPRET \LABELIFY \LABELIFYLIST \NULL \IN \NIN \PREFIX-OF \NOTPREFIXOF \SUFFIXOF \NOTSUFFIXOF \LC \UC \FIRSTLC \FIRSTUC \TITLE.

For example, with

\renewcommand{\glsxtrresourceinit}{% 
\GlsXtrResourceInitEscSequences 
}%

then the earlier example can be written more compactly as:

sort-replace={{{[a-zA-Z]\./}??1}}
A convenient shortcut for use in the `entry-type-aliases` setting:

\GlsXtrBibTeXEntryAliases

This provides aliases for \LaTeX{}'s standard entry types (such as `@article` and `@book`) to `bib2gls`'s `@bibtexentry` entry type (requires `bib2gls v1.4+`).

You may also want to provide storage keys for \LaTeX{}'s standard fields rather than having to alias them all. This can be done with:

\GlsXtrProvideBibTeXFields

This defines each \LaTeX{} field, such as `author`, as a glossary entry key:

\begin{verbatim}
\glsaddstoragekey{address}{}\{\glsxtrbibaddress\}%
\glsaddstoragekey{author}{}\{\glsxtrbibauthor\}%
\glsaddstoragekey{booktitle}{}\{\glsxtrbibbooktitle\}%
\glsaddstoragekey{chapter}{}\{\glsxtrbibchapter\}%
\glsaddstoragekey{edition}{}\{\glsxtrbibedition\}%
\glsaddstoragekey{howpublished}{}\{\glsxtrbibhowpublished\}%
\glsaddstoragekey{institution}{}\{\glsxtrbibinstitution\}%
\glsaddstoragekey{journal}{}\{\glsxtrbibjournal\}%
\glsaddstoragekey{month}{}\{\glsxtrbibmonth\}%
\glsaddstoragekey{note}{}\{\glsxtrbibnote\}%
\glsaddstoragekey{organization}{}\{\glsxtrbiborganization\}%
\glsaddstoragekey{pages}{}\{\glsxtrbibpages\}%
\glsaddstoragekey{publisher}{}\{\glsxtrbibpublisher\}%
\glsaddstoragekey{school}{}\{\glsxtrbibschool\}%
\glsaddstoragekey{series}{}\{\glsxtrbibseries\}%
\glsaddstoragekey{title}{}\{\glsxtrbibtitle\}%
\glsaddstoragekey{bibtextype}{}\{\glsxtrbibtype\}%
\glsaddstoragekey{volume}{}\{\glsxtrbibvolume\}%
\end{verbatim}

This command should be placed before the first `\GlsXtrLoadResources`.

\LaTeX{}'s `type` field clashes with the glossaries package's `type` key, so this command provides the key `bibtextype` instead. You can alias it with `field-aliases =\{type=bibtextype\}` in the resource options.
11.5.2.1. Custom Sort

There are many locale alphabetical rules provided with bib2gls, such as `sort=de-1996` for German new orthography. However, it may be that your particular locale isn’t supported, or you want a rule that covers multiple scripts or non-alphabetic symbols.

The `sort=custom` setting combined with `sort-rule` provides a way to define your own sort rule. For example, suppose I have a file called `animals.bib` that contains:

```latex
@index{bee}
@index{lion}
@index{ant}
@index{cow}
@index{goose}
@index{zebu}
@index{egret}
@index{elk}
@index{llama}
@index{lynx}
@index{bat}
```

Here’s a very limited rule that only recognises five letters:

```latex
\usepackage[record,nostyles,stylemods=bookindex,style=bookindex]{glossaries-extra}
\newcommand{\bibglssetlastgrouptitle}[2]{% \glsxtrsetgrouptitle{#1#2}{Other} \}
\GlsXtrLoadResources[src=animals,selection=all, sort=custom,sort-rule={ < a,A < b,B < e,E < l,L < ll,Ll,LL < z,Z}]
\begin{document}
\printunsrtglossaries
\end{document}
```

Any characters that aren’t included in the rule (such as “c” and “g”) are placed at the end. I’ve defined \bibglssetlastgrouptitle to label that final group of characters “Other”. If the document is in a file called `myDoc.tex`, the build process is:
Example 157: Using \texttt{bib2gls}: simple custom sort rule

\begin{center}
\begin{tabular}{ll}
A & lynx \\
ant & \\
B & llama \\
bat & \\
bee & \\
E & zebu \\
elk & Other \\
egret & \\
L & cow \\
lion & goose
\end{tabular}
\end{center}

Note that “egret” has been placed after “elk”. This is because “l” is included in the rule but “g” isn’t. Whereas “lynx” comes before “llama” because there’s a separate “ll” group after the “l” group.

The commands listed below provide common rule blocks for use in the sort-rule resource option. If you want a rule for a specific locale, you can provide similar commands in a file called \texttt{glossariesxtr-⟨tag⟩.ldf}, where ⟨tag⟩ identifies the dialect, locale, region or root language. See the description of \texttt{\textbackslash IfTrackedLanguageFileExists} in the tracklang documentation for further details. If this file is on \TeX{}’s path and the tracklang package (automatically loaded by glossaries) detects that the document has requested that language or locale, then the file will automatically be loaded. For example, if you want to provide a rule block for Welsh, then create a file called \texttt{glossariesxtr-welsh.ldf} that contains:

\begin{verbatim}
\ProvidesGlossariesExtraLang{welsh}[2018/02/23 v1.0]
\@ifpackageloaded{glossaries-extra-bib2gls}
\end{verbatim}
11. bib2gls: Managing Reference Databases

\{\newcommand{\glsxtrWelshRules}{% 
  \glsxtrLatinA \\
  \string<b,B \\
  \string<c,C \\
  \string<ch,CH \\
  \string<d,D \\
  \string<dd,DD \\
  \string<\glsxtrLatinE \\
  \string<f,F \\
  \string<ff,FF \\
  \string<g,G \\
  \string<ng,NG \\
  \string<\glsxtrLatinH \\
  \string<\glsxtrLatinI \\
  \string<j,J \\
  \string<ll,Ll,LL \\
  \string<\glsxtrLatinM \\
  \string<\glsxtrLatinN \\
  \string<\glsxtrLatinO \\
  \string<\glsxtrLatinP \\
  \string<ph,PH \\
  \string<r,R \\
  \string<rh,RH \\
  \string<\glsxtrLatinS \\
  \string<\glsxtrLatinT \\
  \string<th,TH \\
  \string<u,U \\
  \string<w,W \\
  \string<y,Y 
} % glossaries-extra-bib2gls.sty not loaded
\}

(The use of \string is in case the < character has been made active.) You can provide more than one rule-block per local, to allow for loanwords or foreign words. For example, you could provide \glsxtrWelshIRules, \glsxtrWelshIIRules etc.

If the rules are for a particular script (independent of language or region) then they can be provided in a file given by glossariesxtr-⟨script⟩.ldf instead. For example, the file glossariesxtr-Cyrl.ldf could contain:
Remember that the required document language scripts need to be tracked through the tracklang package, in order for these files to be automatically loaded. This essentially means ensuring you load the appropriate language package before tracklang is loaded by the base glossaries package or any other package that uses it. See the tracklang documentation for further details.

Alternatively, if the rules are specific to a subject rather than a region or language, then you can provide a supplementary package. For example, if you have a package called, say, mapsymbols that provides map symbols, then the file mapsymbols.sty might contain:

```latex
\ProvidesPackage{mapsymbols}
% some package or font loading stuff here to provide
% the appropriate symbols
\newcommand{\Stadium}{...}
\newcommand{\Battlefield}{...}
\newcommand{\Harbour}{...}
% etc

% Provide a rule block:
\newcommand{\MapSymbolOrder}{%
  \glshex 2694 \% crossed-swords 0x2694
  \string< \glshex 2693 \% anchor 0x2693
  \string< \glshex 26BD \% football 0x26BD
}
```

and the supplementary file mapsymbols.bib can provide the appropriate definitions for bib2gls:

```latex
@preamble{"\glsxtrprovidecommand{\Harbour}{\char"2693}
\glsxtrprovidecommand{\Battlefield}{\char"2694}
\glsxtrprovidecommand{\Stadium}{\char"26BD"}"
```

Now both the preamble and rule block can be used in the resource set:
\usepackage{mapsymbols}% my custom package
\usepackage[record]{glossaries-extra}

\GlsXtrLoadResources[
  src={mapsymbols,% \<--- my custom mapsymbols.bib
  entries% data in entries.bib
  },
  sort=custom,
  sort-rule={\glsxtrcontrolrules % control codes
  ;\glsxtrspacerules % space characters
  ;\glsxtrnonprintablerules % non-printable characters
  ;\glsxtrcombiningdiacriticrules % combining diacritics
  ,\glsxtrhyphenrules % hyphens
  <\glsxtrgeneralpuncrules % general punctuation
  <\glsxtrdigitrules % 0, ..., 9
  <\glsxtrfractionrules % fraction symbols
  <\MapSymbolOrder % \<--- custom map symbols
  <\glsxtrMathItalicGreekIrules % math-greek symbols
  <\glsxtrGeneralLatinIrules % Latin letters
  }
]

(As before, you may need to use \string in front of characters like < if they have been made active.)

The following commands are provided by glossaries-extra-bib2gls. They should be separated by the rule separator characters ; (semi-colon) or , (comma) or & (ampersand) or < (less than). See Java’s RuleBasedCollator documentation for details of the rule syntax.

For example, the following will place the mathematical Greek symbols (\alpha, \Alpha, \beta, \Beta etc) in a block before Latin characters:

sort-rule={\glsxtrcontrolrules
 ;\glsxtrspacerules
 ;\glsxtrnonprintablerules
 ;\glsxtrcombiningdiacriticrules
 ,\glsxtrhyphenrules
 <\glsxtrgeneralpuncrules
 <\glsxtrdigitrules
 <\glsxtrfractionrules
 <\glsxtrMathItalicGreekIrules
 <\glsxtrGeneralLatinIVrules
}
11.5.2.1.1. Non-Letters

\texttt{\textbackslash glsxtcontrolrules}

These are control characters that are usually placed at the start of a rule in the ignored section. They typically won’t occur in any sort values, but if they do they should normally be ignored.

\texttt{\textbackslash glsxtspacerules}

These are space characters. They typically come after the control characters with the two blocks separated by a ; (semi-colon).

\texttt{\textbackslash glsxtnonprintablerules}

These are non-printable characters (BOM, tabs, line feed and carriage return). They typically come after the spaces separated by a ; (semi-colon). These characters aren’t checked for by \texttt{bib2gls} when it determines whether or not to use the interpreter, so a TAB or newline character may end up in the sort value if it wasn’t interpreted.

\texttt{\textbackslash glsxtcombiningdiacriticrules}

These are combining diacritic marks which typically follow the space and non-printable blocks (separated by a semi-colon). This command is defined in terms of sub-block commands:

\texttt{\textbackslash newcommand*{\textbackslash glsxtcombiningdiacriticrules}{\%\textbackslash glsxtcombiningdiacriticIrules\string;\textbackslash glsxtcombiningdiacriticIIRules\string;\textbackslash glsxtcombiningdiacriticIIIRules\string;\textbackslash glsxtcombiningdiacriticIVrules\string;}}}
If you prefer, you can use the sub-blocks directly in your required ordered.

\glsxtrcombiningdiacriticIrules

This contains the combining diacritics: acute, grave, breve, circumflex, caron, ring, vertical line above, diaeresis (umlaut), double acute, tilde, dot above, combining macron.

\glsxtrcombiningdiacriticIIRules

This contains the combining diacritics: short solidus overlay, cedilla, ogonek, dot below, low line, overline, hook above, double vertical line above, double grave accent, candrabindu, inverted breve, turned comma above, comma above, reversed comma above, comma above right, grave accent below, acute accent below.

\glsxtrcombiningdiacriticIIIrules

This contains the combining diacritics: left tack below, right tack below, left angle above, horn, left half ring below, up tack below, down tack below, plus sign below, minus sign below, palatalized hook below, retroflex hook below, diaeresis below, ring below, comma below, vertical line below, bridge below, inverted double arch below, caron below, circumflex accent below, breve below, inverted breve below, tilde below, macron below, double low line, tilde overlay, short stroke overlay, long stroke overlay, long solidus overlay, right half ring below, inverted bridge below, square below, seagull below, x above, vertical tilde, double overline, Greek perispomeni, Greek dialytika tonos, Greek ypogeogrammeni, double tilde, double inverted breve, Cyrillic titlo, Cyrillic palatalization, Cyrillic dasia pneumata, Cyrillic psili pneumata.

\glsxtrcombiningdiacriticIVrules

This contains the combining diacritics: left harpoon above, right harpoon above, long vertical line overlay, short vertical line overlay, anticlockwise arrow above, clockwise arrow above, left arrow above, right arrow above, ring overlay, clockwise ring overlay, anticlockwise ring overlay, three dots above, four dots above, enclosing circle, enclosing square, enclosing diamond, enclosing circle backslash, left right arrow above.

\glsxtrhyphenrules

This contains hyphens (including the minus sign 0x2212). This rule block typically comes

569
after the diacritic rules separated by a comma.

\glsxtrgeneralpuncrules

This contains punctuation characters. This rule block typically comes after the hyphen rules separated by a less than (&lt;). As with the combining diacritics, this command is defined in terms of sub-blocks which may be used directly instead if a different order is required:

\newcommand*{\glsxtrgeneralpuncrules}{%
  \glsxtrgeneralpuncIrules
  \string<\glsxtrcurrencyrules
  \string<\glsxtrgeneralpuncIIrules
}\}

\glsxtrgeneralpuncIrules

This is defined as:

\newcommand*{\glsxtrgeneralpuncIrules}{%
  \glsxtrgeneralpuncmarksrules
  \string<\glsxtrgeneralpuncaccentsrules
  \string<\glsxtrgeneralpuncquoterules
  \string<\glsxtrgeneralpuncbracketrules
  \string<\glsxtrgeneralpuncsignrules
}\}

\glsxtrgeneralpuncmarksrules

This contains: underscore, macron, comma, semi-colon, colon, exclamation mark, inverted exclamation mark, question mark, inverted question mark, solidus, full stop.

\glsxtrgeneralpuncaccentsrules

This contains: acute accent, grave accent, circumflex accent, diaeresis, tild, middle dot,
cedilla.

\glsxtrgeneralpuncquoterules
This contains: straight apostrophe, straight double quote, left guillemet, right guillemet.

\glsxtrgeneralpuncbracketrules
This contains: left parenthesis, right parenthesis, left square bracket, right square bracket, left curly bracket, right curly bracket.

\glsxtrgeneralpuncsignrules
This contains: section sign, pilcrow sign, copyright sign, registered sign, at sign.

\glsxtrcurrencyrules
This sub-block contains some currency symbols: currency sign, Thai currency symbol baht, cent sign, colon sign, cruzeiro sign, dollar sign, dong sign, euro sign, French franc sign, lira sign, mill sign, naira sign, peseta sign, pound sign, rupee sign, new sheqel sign, won sign, yen sign.

\glsxtrgeneralpuncIIrules
This sub-block contains some other punctuation symbols: asterisk, backslash, ampersand, hash sign, percent sign, plus sign, plus-minus sign, division sign, multiplication sign, less-than sign, equals sign, greater-than sign, not sign, vertical bar (pipe), broken bar, degree sign, micron sign.

\glsxtrdigitrules
This rule block contains the Basic Latin digits (0, ..., 9) and the subscript and superscript digits (0^o etc) made equivalent to the corresponding Basic Latin digit. The digit block typically comes after the punctuation rules separated by a less than (<).

\glsxtrBasicDigitrules
This rule block contains just the Basic Latin digits (0, ..., 9).

\glsxtrSubScriptDigitrules

This rule block contains just the subscript digits (₀ ... ₉).

\glsxtrSuperScriptDigitrules

This rule block contains just the superscript digits (⁰ ... ⁹).

\glsxtrfractionrules

This rule block contains vulgar fraction characters from the Unicode Number Forms block. The digit block typically comes after the digit rules separated by a less than (<).

\glsxtrIgnorableRules

A shortcut that expands to the ignorable rules:

\glsxtrcontrolrules ;\glsxtrspacerules ;\glsxtrnonprintablerules

\glsxtrGeneralInitRules

A shortcut that expands to common initial rules:

\glsxtrIgnorableRules ;\glsxtrcombiningdiacriticrules ;\glsxtrhyphenrules \glsxtrxgeneralpuncrules \glsxtrxdigitrules \glsxtrxfractionrules

Note that this includes the combining diacritic rules, which won’t be appropriate for languages with accented characters.

There are a number of Latin rule blocks. Some of these included extended characters or ligatures (such as β or œ) but they don’t include accented characters. If you require a
Latin rule block that includes accented characters, digraphs, trigraphs or other extended characters, then it’s best to provide similar commands in a `glossariesxtr-(tag).ldf` file for the particular language or region.

### 11.5.2.1.2. Latin Letters

- **\glsxtrGeneralLatinIrules**

This is just the basic (non-extended) Latin alphabet with the superscript and subscript Latin letters (ª ñ etc) treated as the equivalent basic Latin letter. (If you don’t want the subscripts and superscripts included you can redefine \glsxtrLatinA etc to omit them.)

- **\glsxtrGeneralLatinIIrules**

This is like \glsxtrGeneralLatinIrules but it includes eth (Ð) between “D” and “E” and eszett (ß) treated as “ss”.

- **\glsxtrGeneralLatinIIIRules**

This is like \glsxtrGeneralLatinIrules but it includes eth (Ð) between “D” and “E” and eszett (ß) treated as “sz”.

- **\glsxtrGeneralLatinIVrules**

This is like \glsxtrGeneralLatinIrules but it includes eth (Ð) between “D” and “E”, ae-ligature (æ) is treated as “ae”, oe-ligature (œ) is treated as “oe”, eszett (ß) treated as “ss” and thorn (þ) is treated as “th”.

- **\glsxtrGeneralLatinVrules**

This is like \glsxtrGeneralLatinIrules but it includes eth (Ð) between “D” and “E”, eszett (ß) treated as “ss” and thorn (þ) treated as “th”.

- **\glsxtrGeneralLatinVIrules**

This is like \glsxtrGeneralLatinIrules but it includes eth (Ð) between “D” and “E”, eszett (ß) treated as “ss” and thorn (þ) treated as “th”.
(ß) treated as “sz” and thorn (þ) treated as “th”.

\glsxtrGeneralLatinVIIrules

This is like \glsxtrGeneralLatinIrules but it includes ae-ligature (æ) between “A” and “B”, eth (Ð) between “D” and “E”, insular G (Ɣ) instead of “G”, oe-ligature (œ) between “O” and “P”, long s (ſ) equivalent to “s”, thorn (þ) between “T” and “U” and wynn (ƿ) instead of “W”.

\glsxtrGeneralLatinVIIIrules

This is like \glsxtrGeneralLatinIrules but ae-ligature (æ) is treated as “ae”, oe-ligature (œ) is treated as “oe”, eszett (ß) treated as “ss”, thorn (þ) is treated as “th”, Ø is treated as “O” and “Ł” is treated as “L”.

\glsxtrGeneralLatinAtoMrules

A mini-rule subset of General Latin I rules that just covers A–M.

\glsxtrGeneralLatinNtoZrules

A mini-rule subset of General Latin I rules that just covers N–Z.

\glsxtrGeneralLatinAtoGrules

A mini-rule subset of General Latin I rules that just covers A–G.

\glsxtrGeneralLatinHtoMrules

A mini-rule subset of General Latin I rules that just covers H–M.

\glsxtrGeneralLatinNtoSrules

A mini-rule subset of General Latin I rules that just covers N–S.

\glsxtrGeneralLatinTtoZrules

A mini-rule subset of General Latin I rules that just covers T–Z.
A mini-rule subset of General Latin I rules that just covers T–Z.

<table>
<thead>
<tr>
<th>Rule ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsxtrLatinA</td>
<td>A mini-rule that just covers “A” but includes the sub- and superscript A.</td>
</tr>
<tr>
<td>\glsxtrLatinE</td>
<td>A mini-rule that just covers “E” but includes the subscript E.</td>
</tr>
<tr>
<td>\glsxtrLatinH</td>
<td>A mini-rule that just covers “H” but includes the subscript H.</td>
</tr>
<tr>
<td>\glsxtrLatinK</td>
<td>A mini-rule that just covers “K” but includes the subscript K.</td>
</tr>
<tr>
<td>\glsxtrLatinI</td>
<td>A mini-rule that just covers “I” but includes the subscript I.</td>
</tr>
<tr>
<td>\glsxtrLatinM</td>
<td>A mini-rule that just covers “M” but includes the subscript M.</td>
</tr>
<tr>
<td>\glsxtrLatinN</td>
<td>A mini-rule that just covers “N” but includes the sub- and superscript N.</td>
</tr>
<tr>
<td>\glsxtrLatinO</td>
<td>A mini-rule that just covers “O” but includes the sub- and superscript O.</td>
</tr>
<tr>
<td>\glsxtrLatinP</td>
<td></td>
</tr>
</tbody>
</table>

575
A mini-rule that just covers “P” but includes the subscript P.

\glsxtrLatinP

A mini-rule that just covers “S” but includes the subscript S.

\glsxtrLatinS

A mini-rule that just covers “T” but includes the subscript T.

\glsxtrLatinT

A mini-rule that just covers “X” but includes the subscript X.

\glsxtrLatinX

A mini-rule that just covers eszett (ß) and makes “ſ” (long s followed by short s) equivalent to “ß”. (This is used in the above blocks that treat “ß” as “ss”.)

\glsxtrLatinEszettS

A mini-rule that just covers eszett (ß) and makes “ſz” (long s followed by z) equivalent to “ß”. (This is used in the above blocks that treat “ß” as “sz”.)

\glsxtrLatinEszettS

A mini-rule for eth (Ð, ð) so you don’t need to remember the Unicode values.

\glsxtrLatinEth

A mini-rule for thorn (Þ, þ) so you don’t need to remember the Unicode values.

\glsxtrLatinThorn

\glsxtrLatinAEligature
A mini-rule for ae-ligature (Æ, æ) so you don’t need to remember the Unicode values.

\texttt{\textbackslash glsxtrLatinOELigature}

A mini-rule for oe-ligature (Œ, œ) so you don’t need to remember the Unicode values.

\texttt{\textbackslash glsxtrLatinOslash}

A mini-rule for o-slash (Ø, ø) so you don’t need to remember the Unicode values.

\texttt{\textbackslash glsxtrLatinLslash}

A mini-rule for i-slash (Ł, ł) so you don’t need to remember the Unicode values.

\texttt{\textbackslash glsxtrLatinWynn}

A mini-rule for wynn (Ƿ, ƿ) so you don’t need to remember the Unicode values.

\texttt{\textbackslash glsxtrLatinInsularG}

A mini-rule for insular-G (ӄ, Ʉ) so you don’t need to remember the Unicode values.

\texttt{\textbackslash glsxtrLatinSchwa}

A mini-rule for schwa (Ə, ə, ə) so you don’t need to remember the Unicode values. (Not used in any of the provided Latin rule blocks described above.)

\texttt{\textbackslash glsxtrLatinAA}

A mini-rule for “a with ring above” (Å, å) so you don’t need to remember the Unicode values. (Not used in any of the provided Latin rule blocks described above.)

11.5.2.1.3. Math Greek

\texttt{\textbackslash glsxtrMathGreekIrules}

A rule block for mathematical Greek (\texttt{\alpha}, \texttt{\beta} etc) and upright Greek (\texttt{\upalpha}, \texttt{\upbeta} etc, from the upgreek package) characters that includes digamma (\texttt{\digamma} and \texttt{\Digamma})
between epsilon and zeta. The upright and italic versions are gathered together into the same letter group.

\glsxtrMathGreekIIrules

As \glsxtrMathGreekIrules but doesn’t include digamma.

\glsxtrMathUpGreekIrules

A rule block for upright Greek (\upalpha, etc, from the upgreek package) characters that includes digamma (\digamma and \Digamma) between epsilon and zeta.

\glsxtrMathUpGreekIIrules

A rule block for upright Greek (\upalpha, etc, from the upgreek package) that doesn’t include digamma.

\glsxtrMathItalicGreekIrules

A rule block for mathematical Greek (\alpha, \Alpha, etc) characters that includes digamma (\digamma and \Digamma) between epsilon and zeta. Note that even though the uppercase \Delta etc are actually rendered upright by \LaTeX, \bib2gls’s interpreter treats them as italic to help keep them close to the lowercase versions.

\glsxtrMathItalicGreekIIrules

A rule block for mathematical Greek (\alpha, \Alpha, etc) characters that doesn’t include digamma.

\glsxtrMathItalicUpperGreekIrules

A rule block for uppercase mathematical Greek (\Alpha, \Beta, etc) characters that includes digamma (\Digamma) between epsilon and zeta.

\glsxtrMathItalicUpperGreekIIrules
A rule block for uppercase mathematical Greek (\Alpha, \Beta, etc) characters that doesn’t include digamma.

\glsxtrMathItalicLowerGreekIrules

A rule block for lowercase mathematical Greek (\alpha, \beta, etc) characters that includes digamma (\digamma) between epsilon and zeta.

\glsxtrMathItalicLowerGreekIIrules

A rule block for lowercase mathematical Greek (\alpha, \beta, etc) characters that doesn’t include digamma.

Additionally, there are commands that just cover the uppercase and lowercase forms of a special Greek character (\Upalpha, \upalpha etc and \Alpha, \alpha etc): \glsxtrUpAlpha, \glsxtrMathItalicAlpha, \glsxtrUpBeta, \glsxtrMathItalicBeta, \glsxtrUpGamma, \glsxtrMathItalicGamma, \glsxtrUpDelta, \glsxtrMathItalicDelta, \glsxtrUpEpsilon, \glsxtrMathItalicEpsilon, \glsxtrUpDigamma, \glsxtrMathItalicDigamma, \glsxtrUpZeta, \glsxtrMathItalicZeta, \glsxtrUpEta, \glsxtrMathItalicEta, \glsxtrUpTheta, \glsxtrMathItalicTheta, \glsxtrUpIota, \glsxtrMathItalicIota, \glsxtrUpKappa, \glsxtrMathItalicKappa, \glsxtrUpLambda, \glsxtrMathItalicLambda, \glsxtrUpMu, \glsxtrMathItalicMu, \glsxtrUpNu, \glsxtrMathItalicNu, \glsxtrUpXi, \glsxtrMathItalicXi, \glsxtrUpOmicron, \glsxtrMathItalicOmicron, \glsxtrUpPi, \glsxtrMathItalicPi, \glsxtrUpRho, \glsxtrMathItalicRho, \glsxtrUpSigma, \glsxtrMathItalicSigma, \glsxtrUpTau, \glsxtrMathItalicTau, \glsxtrUpUpsilon, \glsxtrMathItalicUpsilon, \glsxtrUpPhi, \glsxtrMathItalicPhi, \glsxtrUpChi, \glsxtrMathItalicChi, \glsxtrUpPsi, \glsxtrMathItalicPsi, \glsxtrUpOmega, and \glsxtrMathItalicOmega.

Additionally, there are commands for math italic partial differential \(\partial\) (0x1D715) \glsxtrMathItalicPartial and nabla \(\nabla\) (0x1D6FB) \glsxtrMathItalicNabla.

11.5.3. Commands Used Within Resource Files

A glstex file will typically start with \glsnoexpandfields, since supporting field expansion is only necessary where entry definitions may be programmatically performed within a loop or some other command. (This may be suppressed.)

If you use the set-widest resource option, bib2gls v1.8+ will write the following command in the glstex file:

\glsxtrSetWidest{\(\text{type}\)}{\(\text{level}\)}
This sets the widest name for the given glossary type and hierarchical level. This supports both the alttree style and the styles provided by glossary-longextra, which need to know the widest name.

If bib2gls can’t determine the widest name (typically because the name field consists of commands that aren’t recognised by the interpreter) then bib2gls v1.8+ will write the following to the glstex file:

\glsxtrSetWidestFallback{⟨type⟩}{⟨level⟩}{max depth}{list}

Currently the maximum hierarchical depth ⟨max depth⟩ may only be 0 or 2. This command requires commands provided by the glossaries-extra-stylemods package with the alttree style enabled. In this case, it may be simpler to just use \glssetwidest.

The glstex files may also contain instances of \glsxtrprovidestoragekey to provide new glossary entry keys or \provideignoredglossary to provide glossaries to ensure they are defined.

The \bibgls… commands provided by bib2gls are defined in the glstex files with \providecommand. See the bib2gls manual for further details of those commands.

### 11.5.4. Hierarchy

The parent key identifies a parent entry (by its label), so determining if an entry has a parent can easily be achieved by testing if the parent field has been set (using \ifglshasparent or a command like \glsxtrifhasfield).

The other way around, testing if an entry has any children, is much harder. The base glossaries package provides \ifglshaschildren, which is very inefficient as it has to iterate over all entries in the entry’s glossary to determine which ones have a matching parent field.

It’s much easier with bib2gls, as there are resource options available to save this information. The save-child-count option saves the child count for each entry in the childcount internal field. This total only includes the child entries that have been selected by the resource set.

\GlsXtrIfHasNonZeroChildCount{⟨entry-label⟩}{⟨true⟩}{⟨false⟩} modifier: *

A shortcut that uses \GlsXtrIfFieldNonZero to test if the value supplied in the childcount field is non-zero. The starred form uses the starred form of \GlsXtrIfFieldNonZero.

### 11.5.5. Supplemental Locations

Locations in external documents can be manually added by explicitly setting the value when an entry is indexed. However, this is a bit inconvenient so bib2gls provides a way to do this automatically. Both the main document and the supplemental document need to use the record option, and the entries provided via src in the main document must have the same
labels as those in the supplementary document. The supplemental document is identified by the `supplemental-locations` resource option. See the bib2gls manual for further details.

\begin{verbatim}
\glsxtrdisplaysupploc{(prefix)}{(counter)}{(format)}{(src)}{(location)}
modifier: *
\end{verbatim}

This is used by bib2gls version v1.7+ for supplemental locations, instead of using `\glsxtrsupphypernumber` with the `externallocation` attribute. This command sets up the location counter and prefix (used in the formation of hyperlinks) and then uses:

\begin{verbatim}
\glsxtrmultisupplocation{(src)}{(location)}{(format)}
modifier: *
\end{verbatim}

to format the actual location (with an external hyperlink, if supported). The `(format)` (the original location encap) is ignored by default.

### 11.5.6. Nameref Records

Nameref records include the current title and hypertarget. These records are enabled with `record=nameref` and provide a more reliable way of saving the location hypertarget, which can’t be obtained with `makeindex` or `xindy`.

This option is best used with `counter=chapter` or `counter=section` if you want the title included in the location list. If the indexing counter is the default page, only the location number is shown. Similarly for `counter=equation` (or `equations=true`).

Normally locations are recorded in the aux file in the form:

\begin{verbatim}
\glsxtr@record{(entry-label)}{(h-prefix)}{(counter)}{(encap)}{(location)}
\end{verbatim}

This is similar to the command used with `\makenoidxglossaries` and provides the same information about the location and how to form the hypertarget as is passed to `makeindex` and `xindy`. The `record=nameref` option, which requires at least bib2gls v1.8, instead uses:

\begin{verbatim}
\glsxtr@record@nameref{(entry-label)}{(h-prefix)}{(counter)}{(encap)}{(location)}{(current title)}{(current anchor)}{(the-h-counter)}
\end{verbatim}

where `(title)` is obtained from `\@currentlabelname` and `(href)` is obtained from `\@current-Href`. These commands require `hyperref`. If they are undefined, `(title)` and `(href)` will be left empty and bib2gls will treat it as a regular record.
Be careful with this option as \(\langle\text{href}\rangle\) will globally change on every instance of \refstepcounter but \(\langle\text{title}\rangle\) won’t necessarily change. It can therefore cause unexpected behaviour.

The final argument \(\langle\text{hcounter}\rangle\) is obtained from \texttt{\the\texttt{\texttt{counter}}} which provides the partial target name associated with the location counter. With the original makeindex/xindy approach, it’s not possible to include this information in the location, so the base glossaries package attempts to derive a prefix from which the \(\langle\text{hcounter}\rangle\) value can be reconstituted by appending the prefix. Unfortunately, not all definitions of \texttt{\the\texttt{\texttt{counter}}} are in the form \(\langle\text{prefix}\rangle\texttt{\the}\texttt{\texttt{counter}}\) (most notably the equation counter with chapters) so this can fail.

Since \texttt{bib2gls} is customized specifically for use with glossaries-extra, it’s now possible to save \(\langle\text{hcounter}\rangle\), so the \texttt{recordnameref} option does this. By providing both \(\langle\text{href}\rangle\) and \(\langle\text{hcounter}\rangle\), you can determine which target you would rather use. The default is to use \(\langle\text{hcounter}\rangle\), which will take you to the place where the corresponding counter was incremented with \refstepcounter. However, you may choose to switch to using the \(\langle\text{href}\rangle\) target, which will take you to the nearest target before the indexing took place.

With \texttt{bib2gls} v1.8+, normal locations are displayed using:

\[
\glsnoidxdisplayloc\{\langle\text{prefix}\rangle\}\{\langle\text{counter}\rangle\}\{\langle\text{format}\rangle\}\{\langle\text{location}\rangle\}
\]

This is provided by the base glossaries package and is simply defined to do:

\[
\setentrycounter\{\langle\text{prefix}\rangle\}\{\langle\text{counter}\rangle\}\{\texttt{\csuse}\{\langle\text{format}\rangle\}\{\langle\text{location}\rangle\}}
\]

Earlier versions of \texttt{bib2gls} only used this in the \texttt{loclist} field and explicitly used \texttt{\setentrycounter} in the \texttt{location} field followed by \texttt{\{\langle\text{format}\rangle\}\{\langle\text{location}\rangle\}}, which follows the code that’s created with the default \texttt{makeindex} setting. The \texttt{\setentrycounter} command sets up the prefix needed for \texttt{\glshypernumber} to reform the target name from the given location.

The locations identified by \texttt{\glsxtr@record@nameref} are written by \texttt{bib2gls} to the location list using:

\[
\glsxtrdisplaylocnameref\{\langle\text{prefix}\rangle\}\{\langle\text{counter}\rangle\}\{\langle\text{format}\rangle\}\{\langle\text{location}\rangle\}\{\langle\text{title}\rangle\}\{\langle\text{href}\rangle\}\{\langle\text{hcounter}\rangle\}\{\langle\text{file}\rangle\}
\]

With normal internal locations, \(\langle\text{file}\rangle\) will always be empty. With supplemental locations, \(\langle\text{file}\rangle\) will be the external file reference. If \texttt{hyperref} hasn’t been loaded, this command behaves like \texttt{\glsnoidxdisplayloc}, so it will simply encapsulate the location with the given formatting command.

If \texttt{hyperref} has been loaded, then \texttt{\glsxtrdisplaylocnameref} will try to work out the
appropriate hyperlink anchor and text. The \textit{prefix} argument is redundant. It first defines the following commands:

\texttt{\textbackslash glsxtrrecentanchor}

This expands to the \textit{href} argument, which corresponds to the \texttt{\@currentHref} value at the time the location was recorded. If this is used as the anchor, the link will go to the most recent anchor before the record was created. This is more likely to match the given title, but won’t necessarily match the corresponding counter.

For example, if the record was created with \texttt{counter=section} but occurred in a table caption, then \texttt{\glsxtrrecentanchor} and the title will correspond to the table caption. If you have defined \texttt{\glsxtrsectionlocfmt} to show the section number (see below), then this may cause some confusion if clicking on the section number leads to a table caption. However, it will lead to the closest location to where the record was created, which may be preferred.

\texttt{\textbackslash glsxtrlocationanchor}

This expands to the anchor that corresponds to the record’s location counter, which is constructed from the \texttt{counter} and \texttt{hcounter} arguments.

In the above example with the section counter, \texttt{hcounter} will be the value of \texttt{\theHsection} when the record was created, so the constructed anchor will be \texttt{section}. \texttt{hcounter} which corresponds to the anchor at the start of the section. However, if you haven’t defined \texttt{\glsxtrsectionlocfmt}, then the title will correspond to the table caption, which may be a little confusing.

The actual anchor used is obtained from the expansion of:

\texttt{\textbackslash glsxtractualanchor}

This is initialised to the value of \texttt{\glsxtrlocationanchor} but may be changed by:

\texttt{\textbackslash glsxtrsetactualanchor{\langle\textit{counter}\rangle}}

The argument is the counter name, which may be used to set choose an alternative anchor depending on the counter. This command does nothing by default, which means that \texttt{\glsxtrlocationanchor} will be used.

For example, to switch to \texttt{\glsxtrrecentanchor} if the counter is \texttt{page}:
If you are using \texttt{indexcounter} then the recent anchor will be the one created by the \texttt{wrglossary} increment just before the indexing occurs. This means that with \texttt{counter =wrglossary}, the location anchor and recent anchor will be the same. With other counters, the recent anchor will be the closest anchor to the place where indexing occurred.

To allow for different formatting according to the counter name, \texttt{\glsxtrdisplaylocnameref} first checks for the existence of:

\begin{verbatim}
\glsxtr\langle counter\rangle locfmt\{\langle location\rangle\}\{\langle title\rangle\}
\end{verbatim}

If this command is defined, the location will be displayed using:

\begin{verbatim}
\glsxtr\langle format\rangle \glsxtr\langle counter\rangle locfmt\{\langle location\rangle\}\{\langle title\rangle\}
\{\langle href\rangle\}\{\langle file\rangle\}
\end{verbatim}

Note the above warning about the possible mismatch of the title with the anchor. For example, if the following is defined for the section counter:

\begin{verbatim}
\newcommand{\glsxtrsectionlocfmt}[2]{\S#1 (#2)}
\end{verbatim}

then this could lead to the section number followed by the table caption. A more compact form that omits the title is better:

\begin{verbatim}
\newcommand{\glsxtrsectionlocfmt}[2]{\S#1}
\end{verbatim}

The following location formats are predefined. The equation counter:

\begin{verbatim}
\glsxtrequationlocfmt{\langle location\rangle}\{\langle title\rangle\}
\end{verbatim}

This simply expands to \((\langle location\rangle)\) since equations typically don’t have a title, but are usu-
ally enclosed in parentheses.

\glsxtrwrglossaryloctime\{location\}\{title\}

This is used when the \texttt{indexcounter} option creates records with the \texttt{wrglossary} counter. This ensures that the page name is shown rather than the value of the \texttt{wrglossary} counter.

If the corresponding \texttt{\glsxtr\{counter\}loctime} hasn’t been defined, \texttt{\glsxtrdisplaylocnameref} will do one of the following:

- if \langle title \rangle is empty or the counter is \texttt{page}, \texttt{\glsxtrnamereflink} with the title set to the location;
- otherwise it will do:

\glsxtrtitlednamereflink\{format\}\{location\}\{title\}\{file\}

which uses \texttt{\glsxtrnamereflink} with the given \langle title \rangle and \texttt{\glsxtrrecentanchor} as the anchor. This has a better chance of matching the title with the anchor, but it’s not guaranteed as some anchors are created without a title. This is defined as:

\newcommand{\glsxtrtitlednamereflink}[4]{{% \glsxtrnamereflink\{#1\}\{#2\}\{\glsxtrrecentanchor\}\{#4\} %}

This shows the formatted title with the recent anchor. The location isn’t shown. If you would prefer to just show the location and use the anchor corresponding to the location counter:

\renewcommand{\glsxtrtitlednamereflink}[4]{{% \glsxtrnamereflink\{#1\}\{#2\}\{\glsxtrlocationanchor\}\{#3\} %}

If \langle file \rangle is empty an internal link is created with:

\glsxtrfmtinternalnameref\{target\}\{format\}\{file\}

otherwise an external link is created with:

\glsxtrfmtexternalnameref\{target\}\{format\}\{title\}\{file\}
The \textit{file} argument is set by \texttt{bib2gls} for supplemental locations (obtained from \texttt{supplemental -locations}).

The link is encapsulated with the text-block command whose name is given by \texttt{format}, but \texttt{\textbackslash gls\hypernumber} is first locally redefined to \texttt{\@firstofone} to prevent a conflict with the usual location hyperlink formation. This means that if the \texttt{format} is \texttt{\hyperbf} then it will simply behave like \texttt{\textbf{}}.

The following command is provided but not used by default:

\begin{verbatim}
\glsxtrnameloclink\{\texttt{prefix}\}\{\texttt{counter}\}\{\texttt{format}\}\{\texttt{location}\}\{\texttt{text}\}\{\texttt{file}\}
\end{verbatim}

This creates a hyperlink to the target in the given external file obtained from the \texttt{prefix}, \texttt{counter} and \texttt{location} but uses \texttt{text} as the hyperlink text. As with regular indexing, this will fail if the target name can’t be formed by prefixing the location value.

For compactness, \texttt{bib2gls} merges normal records if the \texttt{prefix}, \texttt{counter} and \texttt{location} all match. (An order of precedence can be provided for format conflicts.) With \texttt{nameref} records, you can use the --merge-nameref-on switch provided by \texttt{bib2gls v1.8+} to determine how to merge \texttt{nameref} records. This switch must be followed by one of the following keywords: \texttt{hcounter} (merge on \texttt{hcounter}, default) \texttt{href} (merge on \texttt{href}), \texttt{title} (merge on \texttt{title}) and \texttt{location} (merge on \texttt{location}, as regular records). In all cases, the \texttt{counter} must also match.

\subsection{Dual Entry Commands}

Dual entries can be defined with entry types like \texttt{@dualentry} or \texttt{@dualindexabbreviation}. A single entry definition within the \texttt{bib} file creates two dependent entries that may be referenced within the document and assigned to different glossaries. The default \texttt{selection=} \{recorded and deps\} will ensure that dependent entries are selected, even if they don’t have any records. For example, the following \texttt{bib} entry:

\begin{verbatim}
@dualindexsymbol{L,
   name={Lagrangian function},
   symbol={\ensuremath{L}},
   description={a function of generalized co-ordinates}
}
\end{verbatim}

is essentially like:

\begin{verbatim}
@index{L,
   name={Lagrangian function},
   symbol={\ensuremath{L}},
}
\end{verbatim}
but additionally the two entries (the primary L and the dual dual.L) are dependent. The following document only references the primary entry, but the dependency ensures that the dual is also selected:

```
\usepackage[record,symbols,nostyles,stylemods={tree,bookindex}]{glossaries-extra}
\GlsXtrLoadResources[src=entries,dual-type=symbols]
\glsdefpostlink{index}{\glsxtrpostlinkAddDescOnFirstUse}
\glsdefpostname{index}{\ifglshassymbol{\glscurrententrylabel}{(\glsentrysymbol{\glscurrententrylabel})}{}{}}
\begin{document}
Primary: \gls{L}.
\printunsrtglossary[type=symbols,style=tree]
\printunsrtglossary[title=Index,style=bookindex]
\end{document}
```

The dependency ensures that the dual entry dual.L is selected, and the dual-type setting adds the dual entry to the symbols glossary (which was defined with the symbols package option).

It may be useful to have a hyperlink from the entry in one glossary to its dependent in the other glossary. This can be achieved by instructing bib2gls to save the label of the entry’s opposite using dual-field. The value of this resource option indicates the field in which to save the label. If omitted, dual is assumed.

The glossary style can then be adapted to check if the field has been set and, if so, to create a hyperlink. The following command is provided to assist with this:

```
\GlsXtrDualBackLink{⟨text⟩}{⟨entry-label⟩}
```

This is defined as:
This obtains the field from:

\GlsXtrDualField

\textit{initial: dual}

This expands to dual, which is the default for \texttt{dual-field} if no value is supplied. The above example can be adapted as follows:

\begin{verbatim}
\usepackage[colorlinks]{hyperref}
\usepackage[record,symbols,nostyles,stylemods={tree,bookindex}]{glossaries-extra}
\GlsXtrLoadResources[src=entries, dual-type=symbols,dual-field]
\glsdefpostlink{index}{\glsxtrpostlinkAddSymbolOnFirstUse}
\glsdefpostname{index}{\ifglshassymbol{\glscurrententrylabel}
\GlsXtrDualBackLink{\glssentrysymbol{\glscurrententrylabel}}{}\else\fi}
\begin{document}
Primary: \gls{L}.
\printunsrtglossary[type=symbols,style=tree]
\printunsrtglossary[title=Index,style=bookindex]
\end{document}
\end{verbatim}

Example 158: Using bib2gls: dual backlinks

Primary: Lagrangian function \((L)\).

\textbf{Symbols}

\(L\) (Lagrangian function) a function of generalized co-ordinates

\textbf{Index}

Lagrangian function \((L)\), 1
This checks if the symbol key has been set in both the category post-link hook and the category post-name hook to append the symbol in parentheses. In addition, the category post-name hook adds a link to the corresponding dual entry.

If you prefer instead to have the backlink on the name in both glossaries, then this can more easily be achieved with a resource option such as dual-backlink.

The commands described below were designed for use with bib2gls’s dual entries, but may also be used in other contexts where a label may potentially have a number of possible prefixes.

It’s possible to use commands like \glsxtrnewgls to create \gls-like commands that automatically insert a label prefix (such as dual. for dual entries). The commands described in this section provide a similar set of \gls-like commands that iterate over a set of possible prefixes until a match is found.

Each possible prefix (which may be empty) is identified by:

\begin{itemize}
\item \glsxtraddlabelprefix{⟨label-prefix⟩}
\item \glsxtrprependlabelprefix{⟨label-prefix⟩}
\item \glsxtrclearlabelprefixes
\end{itemize}

These should be listed in order of precedence.

You can prepend a prefix to the list using:

\begin{itemize}
\item \glsxtraddlabelprefix{⟨label-prefix⟩}
\item \glsxtrprependlabelprefix{⟨label-prefix⟩}
\item \glsxtrclearlabelprefixes
\end{itemize}

which gives it the highest order of precedence.

The list of known prefixes can be (locally) cleared with:

\begin{itemize}
\item \glsxtraddlabelprefix{⟨label-prefix⟩}
\item \glsxtrprependlabelprefix{⟨label-prefix⟩}
\item \glsxtrclearlabelprefixes
\end{itemize}

You can test if a prefix is already in the list with:

\begin{itemize}
\item \glsxtraddlabelprefix{⟨label-prefix⟩}
\item \glsxtrprependlabelprefix{⟨label-prefix⟩}
\item \glsxtrclearlabelprefixes
\end{itemize}

This does \texttt{true} if the given prefix has been added to the list. No expansion is performed on \langle label-prefix \rangle.

In the event that there’s no match for any of the prefixes (which will occur on the first \LaTeX run before the \texttt{glistex} file has been created), the fallback is determined by the conditional:

\begin{itemize}
\item \glsxtraddlabelprefix{⟨label-prefix⟩}
\item \glsxtrprependlabelprefix{⟨label-prefix⟩}
\item \glsxtrclearlabelprefixes
\end{itemize}

\texttt{\ifGlsXtrPrefixLabelFallbackLast \texttt{true} \else \texttt{false} \fi initial: \iftrue
If true, this will fallback on the last prefix, otherwise it will fallback on the first. This conditional can be set to true with:

\GlsXtrPrefixLabelFallbackLasttrue

and to false with:

\GlsXtrPrefixLabelFallbackLastfalse

The default is true.

As from v1.49, all possible matches will be recorded using a special syntax if none are found. This ensures there will be at least one match on the first run. However, note that this requires \texttt{bib2gls v3.0+}.

In general it’s best to avoid adding multiple instances of the same prefix, so you can check with this command before adding a prefix to the list. However, it can be useful to repeat a prefix at the start or end of the list so that it can be used as a fallback for entries that haven’t yet been defined.

With the list of possible prefixes set up (including an empty prefix if necessary), you can use:

\dgls[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩

which behaves like

\gls[⟨options⟩]{⟨prefix⟩{⟨entry-label⟩}}[⟨insert⟩]

where ⟨prefix⟩ is the first prefix in the list such that ⟨prefix⟩⟨label⟩ matches a defined entry. This requires \texttt{bib2gls v3.0+} to work properly on the first \LaTeX run (when no entries are defined).

There are also analogous commands for the plural and case-changing versions:

\dglspl[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩

\glspl[⟨options⟩]{⟨prefix⟩{⟨entry-label⟩}}[⟨insert⟩]
11. bib2gls: Managing Reference Databases

(uses \glsp),

\dGls\{⟨options⟩\}{⟨entry-label⟩}{⟨insert⟩}
  modifiers: * + ⟨alt-mod⟩

(uses \Gls),

\dGls\{⟨options⟩\}{⟨entry-label⟩}{⟨insert⟩}
  modifiers: * + ⟨alt-mod⟩

(uses \Glsp),

\dGLS\{⟨options⟩\}{⟨entry-label⟩}{⟨insert⟩}
  modifiers: * + ⟨alt-mod⟩

(uses \GLS),

\dGLS\{⟨options⟩\}{⟨entry-label⟩}{⟨insert⟩}
  modifiers: * + ⟨alt-mod⟩

(uses \GLSp),

\dglslink\{⟨options⟩\}{⟨entry-label⟩}{⟨text⟩}
  modifiers: * + ⟨alt-mod⟩

(uses \glslink),

\dglslink\{⟨options⟩\}{⟨entry-label⟩}{⟨text⟩}
  modifiers: * + ⟨alt-mod⟩

(uses \Glslink),

\dglsdisp\{⟨options⟩\}{⟨entry-label⟩}{⟨text⟩}
  modifiers: * + ⟨alt-mod⟩

(uses \glsdisp) and

\dglsdisp\{⟨options⟩\}{⟨entry-label⟩}{⟨text⟩}
  modifiers: * + ⟨alt-mod⟩

(uses \Glsdisp).
If you want to use a specific field you can instead use:

\dglsfield[(options)]\{entry-label\}\{field-label\}\{text\}

modifiers: * +

This will find the first match that has the given field set (that is, the field value has been defined and is not empty or \relax). The field should be identified by its internal field label. There are also case-changing versions:

\dGlsfield[(options)]\{entry-label\}\{field-label\}\{text\}

which applies a sentence case change and

\dGLSfield[(options)]\{entry-label\}\{field-label\}\{text\}

which applies all caps. Note that at least one of the potential labels must have the given field set in order for the reference to be correctly resolved. For example:

\dglsfield{pi}{symbol}

If you find this a bit long-winded to type and you want to provide a shorter command that recognises the modifiers, then you can use:

\newdglsfield[(default-options)]\{field\}\{cs\}

For example:

\newdgsfield{symbol}{\sym}
\sym{pi}

If you also want sentence case and all caps versions use:

\newdglasfieldlike[(default-options)]\{field\}\{cs\}\{Cs\}\{CS\}
where \(\langle cs\rangle\) is the command without a case-change (which will use \texttt{\dglsf{}}), \(\langle Cs\rangle\) is the sentence case command (which will use \texttt{\dGls{}}) and \(\langle CS\rangle\) is the all caps command (which will use \texttt{\dGLS{}}). This will also use \texttt{\glsmfuaddmap} and \texttt{\glsmfublocker} to establish the sentence case mapping from \(\langle cs\rangle\) to \(\langle Cs\rangle\) and block the case change for \(\langle CS\rangle\).

No expansion is performed on \(\langle default-options\rangle\) or \(\langle field\rangle\) while the new commands are being defined.

Information is written to the transcript file with \texttt{\GlossariesExtraInfo} if a fallback is considered. Note that this shouldn’t be considered a warning as the fallback might be desired, but if the wrong output is produced this information may explain the selection.

This family of \texttt{\dglsf{}} commands all define:

\texttt{\dglsf{currentfieldlabel}}

This expands to the given field label.

If the requested field isn’t set (according to \texttt{\ifcsvoid{}}) then the fallback field will be tried instead.

\texttt{\dglsf{fallbackfieldlabel}}

\texttt{\textit{initial}: text}

This expands to the fallback field, which defaults to the \texttt{text} field.

If you need to know whether the requested field or the fallback field was used, the following will be set to the actual field used.

\texttt{\dglsf{actualfieldlabel}}

For example, suppose the file \texttt{entries.bib} contains:

\begin{verbatim}
@index{duck}
@dualindexabbreviation{svm,
    short={SVM},
    long={support vector machine}
}
@dualindexsymbol{pi,
    symbol={\ensuremath{\pi}},
    description=ratio of a circle's circumference to its diameter
}
\end{verbatim}
and suppose the document code is:

```latex
\documentclass{article}
\usepackage{hyperref}
\usepackage[record,abbreviations,symbols]{glossaries-extra}
\newcommand{\bibglsnewdualindexsymbolsecondary}[5]{%
  \longnewglossaryentry*{#1}{name={#3},category={symbol},
  type={symbols},symbol={#4},#2}{#5}%
}
\GlsXtrLoadResources[src=entries]
\begin{document}
First use: \gls{duck}, \gls{svm}, \gls{pi}.
Next use: \gls{duck}, \gls{svm}, \gls{pi}.
\printunsrtglossaries
\end{document}
```

This uses the default empty primary prefix and dual. for the dual prefix, so \gls{svm} is referencing the primary entry, which is (essentially) an @index type not an abbreviation. It therefore doesn’t follow the abbreviation style, and it also hyperlinks to the index not to the list of abbreviations. Similarly for \gls{pi}, which references the primary @index entry rather than the symbol.

What’s really needed is:

\gls{duck}, \gls{dual.svm}, \gls{dual.pi}.

or use \texttt{label-prefix} and \texttt{dual-prefix} to set the label prefixes:

```latex
\GlsXtrLoadResources[src=entries,
  label-prefix={idx.},dual-prefix={}]
```

then only the entries without a dual need a prefix:

\gls{idx.duck}, \gls{svm}, \gls{pi}.

Using \texttt{\glsxtrnewgls} or \texttt{\glsxtrnewglslike} (see §5.7) to define the custom \idx:
the entry references can be simplified to:

\idx{duck}, \gls{svm}, \gls{pi}.

but this requires remembering which terms have duals. An alternative is to use \dgls instead:

\GlsXtrLoadResources[src=entries, combine-dual-locations=primary]
\glsxtraddlabelprefix{dual.}
\glsxtraddlabelprefix{}

\begin{document}
First use: \dgls{duck}, \dgls{svm}, \dgls{pi}.
Next use: \dgls{duck}, \dgls{svm}, \dgls{pi}.
\printunsrtglossaries
\end{document}
Example 159: Using \texttt{bib2gls}: dual entry label prefixes

First use: duck, support vector machine (SVM), $\pi$. Next use: duck, SVM, $\pi$.

\textbf{Glossary}

\begin{itemize}
\item \textsl{duck} 1
\item \textsl{pi} ratio of a circle’s circumference to its diameter 1
\item \textsl{SVM} 1
\end{itemize}

\textbf{Symbols}

\begin{itemize}
\item $\pi$ ratio of a circle’s circumference to its diameter
\end{itemize}

\textbf{Abbreviations}

\begin{itemize}
\item \textsl{SVM} support vector machine
\end{itemize}

On the first \LaTeX{} call (when the glstex file doesn’t exist), neither dual.svm nor svm exists, so \texttt{dgl{s}} uses the last prefix (which is empty in this case). This means that on the first \LaTeX{} run, \texttt{dgl{s}{svm}} behaves like \texttt{gls{svm}}, which adds a record for the primary svm entry. The default primary-dual dependency means that this will cause both the primary (svm) and dual (dual.svm) entry to be selected. The location will be added to the primary entry’s location list, unless overridden by resource options, such as \texttt{combine-dual-locations}.

Once \texttt{bib2gls} has been run and the glstex file exists, then dual.svm exists. So \texttt{dgl{s}{svm}} will again first try dual.svm (as dual.svm is the first in the list of label prefixes). That now exists, so \texttt{dgl{s}{svm}} now behaves like \texttt{gl{s}{dual.svm}}, which follows the abbreviation style and hyperlinks to the list of abbreviations. Similarly for the index-symbol combination dual.pi and pi.

In the case of \texttt{dgl{s}{duck}}, which doesn’t have a dual, the label dual.duck never exists, so that’s never selected. However, when there’s no match, such as when the glstex file doesn’t exist, the duck entry will be recorded with both the dual. prefix and the empty prefix. This allows \texttt{bib2gls} to test which prefix-label combination matches.
If you haven’t used `combine-dual-locations` an extra `bib2gls`+\LaTeXx run may be required to correct the location lists.

If you change the label prefixes, remember to update the corresponding \glsxtraddlabelprefix\{⟨label-prefix⟩\}. If no prefixes have been added to the list (or if the list is cleared), just an empty prefix is assumed.

Note that `bib2gls v1.8+` provides hooks that identify the label prefixes in the glstex file:

\begin{verbatim}
\bibglstertiaryprefixlabel\{⟨label-prefix⟩\}
\bibglsdualprefixlabel\{⟨label-prefix⟩\}
\bibglsprimaryprefixlabel\{⟨label-prefix⟩\}
\end{verbatim}

These do nothing by default, but they can be defined before the resource file is loaded to set up the prefix list. For example:

\begin{verbatim}
\newcommand{\bibglsprimaryprefixlabel}[1]\{%
  \glsxtraddlabelprefix{#1}
\}
\newcommand{\bibglsdualprefixlabel}[1]\{%
  \glsxtrprependlabelprefix{#1}
\}
\GlsXtrLoadResources[src=entries]
\end{verbatim}

Remember that this will only have an effect once the glstex file has been created. The prefix list will be empty on the first run (which is treated as a single empty prefix). If this isn’t a suitable fallback, it may be necessary to add one after all the resource commands:

\begin{verbatim}
\newcommand{\bibglsprimaryprefixlabel}[1]\{%
  \glsxtraddlabelprefix{#1}
\}
\newcommand{\bibglsdualprefixlabel}[1]\{%
  \glsxtrprependlabelprefix{#1}
\}
\GlsXtrLoadResources[src=entries,label-prefix={idx.}]
\glsxtraddlabelprefix{idx.}
\end{verbatim}

Although this rather defeats the purpose of using the hooks as you still have to keep track of the fallback prefix.

### 11.5.8. Supplementary Commands

\begin{verbatim}
\glsdisplaynumberlist\{⟨entry-label⟩\}
\end{verbatim}
This command is provided by the base glossaries package, but it requires hooking into the location list formatting within the glossary to obtain the list. This information is more easily available with bib2gls, which stores the formatted location list in the \texttt{location} field, so glossaries-extra-bib2gls redefines this command to use that field value. Likewise for:

\begin{verbatim}
\glsentrynumberlist{(entry-label)}
\end{verbatim}

which is redefined to simply do:

\begin{verbatim}
\glsxtrusefield{#1}{location}
\end{verbatim}

\begin{verbatim}
\IfTeXParserLib{⟨\TeX parser lib code⟩}{⟨\LaTeX code⟩}
\end{verbatim}

This command is defined by glossaries-extra-bib2gls to expand to \LaTeX code, but is defined by the \TeX parser library used by bib2gls’s to expand \TeX parser lib code. May be used in the \texttt{@preamble} or in field values to provide bib2gls with alternative content.

\begin{verbatim}
\glsxtrprovidecommand{⟨cs⟩}[(⟨n⟩)][(⟨default⟩)]{(⟨definition⟩)} modifier: *
\end{verbatim}

This command is intended for use in \texttt{@preamble}. It’s simply defined to \texttt{\providecommand} in glossaries-extra-bib2gls but bib2gls’s interpreter treats it as \texttt{\renewcommand}. This means that you can override bib2gls’s internal definition of a command without overriding the command definition in the document (if it’s already defined before the resource file is input). For example:

\begin{verbatim}
@preamble{"\glsxtrprovidecommand{\int}{integral}"
\end{verbatim}

This will force bib2gls to treat \texttt{\int} as the word “integral” to assist sorting but if this preamble code is written to the \texttt{glstex} file (as it is by default) then it won’t override the current definition (provided by the kernel or redefined by a package).

The helper commands in the resource files are defined using \texttt{\providecommand}. For many of them, if you want to provide an alternative definition then you need to define the command before the resource file is loaded. There are a few that may be redefined afterwards but if you use \texttt{\renewcommand} then you will get an error on the first \LaTeX run when the \texttt{glstex} file doesn’t exist. In this case, you may prefer to use:

\begin{verbatim}
\glsrenewcommand{⟨cs⟩}[(⟨n⟩)][(⟨default⟩)]{(⟨definition⟩)} modifier: *
\end{verbatim}
This behaves like \renewcommand but only generates a warning rather than an error if the command isn’t already defined so it won’t interrupt the document build.

\GlsXtrIndexCounterLink{(text)}{(entry-label)}

For use with the indexcounter package option and the save-index-counter resource option. This command creates a hyperlink (if supported) to the target obtained from the indexcounter field (which will be set in the glstex file, if applicable).

If then indexcounter field is set for the entry given by \textit{(entry-label)}, this command does \hyperref[wrglossary.(value)]{(text)}, where \textit{(value)} is the value of the indexcounter field. If the field isn’t set or if \hyperref hasn’t been defined, this just does \textit{(text)}. See the bib2gls manual (v1.4+) for further details.

The glossaries-extra-bib2gls package also provides definitions of the missing mathematical Greek commands: \texttt{\Alpha}, \texttt{\Beta}, \texttt{\Epsilon}, \texttt{\Zeta}, \texttt{\Iota}, \texttt{\Kappa}, \texttt{\Mu}, \texttt{\Nu}, \texttt{\Omicron}, \texttt{\Rho}, \texttt{\Tau}, \texttt{\Chi}, \texttt{\Digamma}, \texttt{omicron}. These are all defined with \providecommand, so they won’t override any definitions provided by any package loaded before glossaries-extra. Since bib2gls’s interpreter recognises these commands, using them instead of explicitly using the Latin characters with the same shape helps to keep the Greek symbols together when sorting. Similarly, if upgreek has been loaded, the missing upright Greek commands are also provided: \texttt{\Upalpha}, \texttt{\Upbeta}, \texttt{\Upepsilon}, \texttt{\Upzeta}, \texttt{\Upeta}, \texttt{\Upiota}, \texttt{\Upkappa}, \texttt{\Upmu}, \texttt{\Upnu}, \texttt{\Upomicron}, \texttt{\Uprho}, \texttt{\Uptau}, \texttt{\Upchi}, \texttt{upomicron}.
12. Auto-Indexing

It’s possible that you may also want a normal index as well as the glossary, and you may want entries to automatically be added to the index (as in this document). There are two attributes that govern this: `indexname` and `dualindex`.

The auto-indexing is designed for `makeindex` syntax. If you’ve used the `xindy` package option, the automatic escaping of `xindy` special characters in the `sort` field may result in an incorrect sort value for the `\index` command used by the auto-indexing. Note also that `texindy` has a fixed set of special characters (corresponding to `makeindex`s defaults) that can’t be customized. You may want to consider using `bib2gls` and its dual entries as an alternative approach.

The `\glsxtrpostnamehook` macro, used at the end of `\glossentryname` and `\Glossentryname`, checks the `indexname` attribute for the category associated with that entry. Since `\glossentryname` is used in the default glossary styles, this makes a convenient way of automatically indexing each entry name at its location in the glossary without fiddling around with the value of the `name` key.

The internal macro used by the `glossaries` package to write the information to the external glossary file is modified to check for the `dualindex` attribute.

In both cases, the indexing is done through:

\begin{verbatim}
\glsxtrdoautoindexname\{⟨entry-label⟩\}\{⟨attribute⟩\}
\end{verbatim}

This uses the standard `\index` command with the sort value taken from the entry’s `sort` key and the actual value set to `\glossentryname\{⟨entry-label⟩\}`. There are user-level commands available to change the sort and actual value used by the automated index.

The actual value is given by:

\begin{verbatim}
\glsxtrautoindexentry\{⟨entry-label⟩\}
\end{verbatim}

where `⟨entry-label⟩` is the entry’s label. The default definition is:

\begin{verbatim}
\newcommand*{\glsxtrautoindexentry}{\string\glossentryname{♯1}}
\end{verbatim}
12. Auto-Indexing

Note the use of \texttt{\string} to prevent \texttt{\glsentryname} from being expanded as it’s written to the index file.

The sort value is assigned using:

\begin{verbatim}
\glsxtraautoindexassignsort{⟨entry-label⟩}
\end{verbatim}

where ⟨entry-label⟩ is the entry label and ⟨cs⟩ is the command which needs to be set to the sort value. The default definition is:

\begin{verbatim}
\newcommand*{\glsxtraautoindexassignsort}[2]{%
  \glsletentryfield{#1}{#2}{sort}%
}
\end{verbatim}

After this macro is called, ⟨cs⟩ is then processed to escape any of \texttt{makeindex}’s special characters. Note that this escaping is only performed on the sort not on the actual value. The escaping of the sort value is performed by

\begin{verbatim}
\glsxtraautoindexesc
\end{verbatim}

You can redefine this to do nothing if you want to omit the escaping. You may want to consider providing another field to obtain the appropriate sort value if the one provided in the sort field isn’t suitable (because it may already have had special characters escaped or it may be a numeric value in the case of sort by use or definition).

The command used to perform the actual indexing is:

\begin{verbatim}
\glsxtraautoindex
\end{verbatim}

This just does \texttt{\index{⟨text⟩}} by default.

The entry’s parent field isn’t referenced in this automated indexing.

For example, to index the value of the \texttt{first} key, instead of the \texttt{name} key:

\begin{verbatim}
\renewcommand*{\glsxtraautoindexentry}[1]{\string\glsentryfirst{#1}}
\end{verbatim}

and if the sort value also needs to be set to the \texttt{long} field, if present, otherwise the \texttt{sort} field:
12. Auto-Indexing

\renewcommand*{\glsxtrautoindexassignsort}[2]{%
  \ifglshaslong{#2}%
  {\glsletentryfield{#1}{#2}{long}}%
  {\glsletentryfield{#1}{#2}{sort}}%
\}

If the value of the attribute is “true”, no encap will be added, otherwise the encap will be the attribute value. For example:

\glssetcategoryattribute{general}{indexname}{textbf}

will set the encap to \textbf which will display the relevant page number in bold whereas

\glssetcategoryattribute{general}{dualindex}{true}

won’t apply any formatting to the page number in the index.

The location used in the index will always be the page number not the counter used in the glossary. (Unless some other loaded package has modified the definition of \index to use some thing else.)

By default the format key won’t be used with the dualindex attribute. You can allow the format key to override the attribute value by using the preamble-only command:

\GlsXtrEnableIndexFormatOverride

If you use this command and hyperref has been loaded, then the theindex environment will be modified to redefine \glshypernumber to allow formats that use that command.

The dualindex attribute will still be used on subsequent use even if the indexonly-first attribute (or indexonlyfirst package option) is set. However, the dualindex attribute will honour the noindex key.

The \glsxtrdoautoindexname command will attempt to escape any of makeindex’s special characters, but there may be special cases where it fails, so take care. This assumes the default makeindex actual, level, quote and encap values (unless any of the commands \actualchar, \levelchar, \quotechar or \encapchar have been defined before glossaries-extra is loaded).
12. Auto-Indexing

If this isn’t the case, you can use the following preamble-only commands to set the correct characters.

Be very careful of possible shifting category codes!

\GlsXtrSetActualChar{⟨character⟩}

Set the actual character to ⟨char⟩.

\GlsXtrSetLevelChar{⟨character⟩}

Set the level character to ⟨char⟩.

\GlsXtrSetEscChar{⟨character⟩}

Set the escape (quote) character to ⟨char⟩.

\GlsXtrSetEncapChar{⟨character⟩}

Set the encap character to ⟨char⟩.
13. On-the-Fly Document Definitions

The commands described here may superficially look like \index{⟨word⟩}, but they behave rather differently. If you want to use \index then just use \index.

The base glossaries package advises against defining entries in the document environment. As mentioned in §2.4, this ability is disabled by default with glossaries-extra but can be enabled using the docdef package options.

Although this can be problematic, the glossaries-extra package provides a way of defining and using entries within the document environment without the tricks used with the docdef option. There are limitations with this approach, so take care with it. This function is disabled by default, but can be enabled using the preamble-only command:

\GlsXtrEnableOnTheFly

When used, this defines the commands described below. The starred version was provided to help support UTF-8 with inputenc. Recent versions of the \LaTeX kernel now provide better support. Both glossaries and glossaries-extra have been updated to help integrate the new kernel features, so the unstarrred version may also work with UTF-8.

If you use the starred version of \GlsXtrEnableOnTheFly don’t use any commands in the ⟨label⟩, even if they expand to just text.

\glsxtr[⟨gls-options⟩][⟨dfn-options⟩]{⟨entry-label⟩}

If an entry with the label ⟨entry-label⟩ has already been defined, this just does \gls[⟨gls-options⟩]{⟨entry-label⟩}. If ⟨entry-label⟩ hasn’t been defined, this will define the entry using:

\newglossaryentry{⟨entry-label⟩}{name={⟨entry-label⟩}, category=\glsxtrcat, description={\nopostdesc}, ⟨dfn-options⟩}
13. On-the-Fly Document Definitions

The \langle label \rangle must contain any non-expandable commands, such as formatting commands or problematic characters. If the term requires any of these, they must be omitted from the \langle entry-label \rangle and placed in the \texttt{name} key within the optional argument \langle dfn-options \rangle.

The second optional argument \langle dfn-options \rangle should be empty if the entry has already been defined, since it’s too late for them. If it’s not empty, a warning will be generated with:

\GlsXtrWarning\{\langle options \rangle\}\{\langle entry \rangle\}

For example, this warning will be generated on the second instance of \texttt{\glsxtr} below:

\glsxtr[]\{[plural=geese]\{goose\}
\% ... later
\glsxtr[]\{[plural=geese]\{goose\}

If you are considering doing something like:

\newcommand*{\goose}{\glsxtr[]\{[plural={geese}]\{goose\}}
\renewcommand*{\GlsXtrWarning}[2]{}
\% ... later
\goose\ some more text here

then don’t bother. It’s simpler and less problematic to just define the entries in the preamble with \texttt{\newglossaryentry} and then use \texttt{\gls} in the document.

There are also plural and case-changing alternatives to \texttt{\glsxtr}.

\glsxtrpl[\langle gls-options \rangle][\langle dfn-options \rangle]\{\langle entry-label \rangle\}

This is like \texttt{\glsxtr} but uses \texttt{\glspl} instead of \texttt{\gls}.

\Glsxtr[\langle gls-options \rangle][\langle dfn-options \rangle]\{\langle entry-label \rangle\}

This is like \texttt{\glsxtr} but uses \texttt{\Gls} instead of \texttt{\gls}.

\Glsxtrpl[\langle gls-options \rangle][\langle dfn-options \rangle]\{\langle entry-label \rangle\}
This is like \glsxtr but uses \Glspl instead of \gls.

The category is set to:

\glsxtrcat

\textit{initial: general}

This should simply expand to the required category label. The default definition is general.

The commands \glsxtr, \glsxtrpl, \Glsxtr and \Glsxtrpl can’t be used after the glossaries have been displayed (through \printglossary etc). It’s best not to mix these commands with the standard glossary commands, such as \gls or there may be unexpected results.
14. Supplementary Files

The glossaries-extra package comes with some additional files. Those listed in §14.1 provide dummy entries for testing various styles. They should be placed on \TeX’s path.

There are also some sample files listed in §14.2. These should be located in the package documentation directory.

14.1. Dummy Files for Testing

The base glossaries package provides files with dummy entries for testing. The glossaries-extra package provides an additional file with entries.

- example-glossaries-xr.tex
  This file contains entries that have the see, seealso or alias keys set.
  There are also bib files corresponding to all the available tex files for use with bib2gls.

- example-glossaries-acronym.bib
  Corresponds to example-glossaries-acronym.tex

- example-glossaries-acronym-desc.bib
  Corresponds to example-glossaries-acronym-desc.tex

- example-glossaries-acronyms-lang.bib
  Corresponds to example-glossaries-acronyms-lang.tex

- example-glossaries-brief.bib
  Corresponds to example-glossaries-brief.tex

- example-glossaries-childmultipar.bib
  Corresponds to example-glossaries-childmultipar.tex

- example-glossaries-childnoname.bib
  Corresponds to example-glossaries-childnoname.tex

- example-glossaries-cite.bib
  Corresponds to example-glossaries-cite.tex

- example-glossaries-images.bib
  Corresponds to example-glossaries-images.tex

- example-glossaries-long.bib
14. Supplementary Files

Corresponds to example-glossaries-long.tex

example-glossaries-longchild.bib
Corresponds to example-glossaries-longchild.tex

example-glossaries-multipar.bib
Corresponds to example-glossaries-multipar.tex

example-glossaries-parent.bib
Corresponds to example-glossaries-parent.tex

example-glossaries-symbolnames.bib
Corresponds to example-glossaries-symbolnames.tex

example-glossaries-symbols.bib
Corresponds to example-glossaries-symbols.tex

example-glossaries-url.bib
Corresponds to example-glossaries-url.tex

example-glossaries-user.bib
Corresponds to example-glossaries-user.tex

example-glossaries.xr.bib
Corresponds to example-glossaries.xr.tex

14.2. Sample Files

The glossaries-extra package comes with some sample files that are listed below. There are also sample files provided with the glossaries package and with bib2gls. See also the Dickimaw Books Gallery.¹

sample.tex

Simple sample file that uses one of the dummy files provided by the glossaries package for testing.

pdflatex sample
makeglossaries sample
pdflatex sample
pdflatex sample

sample-abbr-styles.tex

¹dickimaw-books.com/gallery
14. Supplementary Files

 Demonstrates all predefined abbreviation styles.  

 sample-mixture.tex  

 General entries, acronyms and initialisms all treated differently.  

 sample-name-font.tex  

 Categories and attributes are used to customize the way different entries appear.  

 sample-abbrv.tex  

 General abbreviations.  

 sample-acronym.tex  

 Acronyms aren’t initialisms and don’t expand on first use.
14. Supplementary Files

- **sample-acronym-desc.tex**
  
  ```
  pdflatex sample-acronym-desc
  makeglossaries sample-acronym-desc
  pdflatex sample-acronym-desc
  makeglossaries sample-acronym-desc
  pdflatex sample-acronym-desc
  ```

  Acronyms that have a separate long form and description.

- **sample-crossref.tex**
  
  ```
  pdflatex sample-crossref
  makeglossaries sample-crossref
  pdflatex sample-crossref
  ```

  Unused entries that have been cross-referenced automatically are added at the end of the document.

- **sample-indexhook.tex**
  
  ```
  pdflatex sample-indexhook
  makeglossaries sample-indexhook
  pdflatex sample-indexhook
  ```

  Use the index hook to track which entries have been indexed (and therefore find out which ones haven’t been indexed).

- **sample-footnote.tex**
  
  ```
  pdflatex sample-footnote
  makeglossaries sample-footnote
  pdflatex sample-footnote
  ```

  Footnote abbreviation style that moves the footnote marker outside of the hyperlink generated by \gls and moves it after certain punctuation characters for neatness.
14. Supplementary Files

```
pdflatex sample-undef
makeglossaries sample-undef
pdflatex sample-undef
pdflatex sample-undef
```
Warn on undefined entries instead of generating an error.

```
pdflatex sample-mixed-abbrv-styles
makeglossaries sample-mixed-abbrv-styles
pdflatex sample-mixed-abbrv-styles
```
Different abbreviation styles for different entries.

```
pdflatex sample-initialisms
makeglossaries sample-initialisms
pdflatex sample-initialisms
```
Automatically insert dots into initialisms.

```
pdflatex sample-postdot
makeglossaries sample-postdot
pdflatex sample-postdot
```
Another initialisms example.

```
pdflatex sample-postlink
makeglossaries sample-postlink
pdflatex sample-postlink
```
Automatically inserting text after the link text produced by commands like `\gls` (outside of hyperlink, if present).

```
pdflatex sample-header
```
14. Supplementary Files

Using entries in section/chapter headings.

```
pdflatex sample-header
makeglossaries sample-header
pdflatex sample-header
```

Using the `dualindex` and `indexname` attributes to automatically add glossary entries to the index (in addition to the glossary location list).

```
pdflatex sample-autoindex
makeglossaries sample-autoindex
pdflatex sample-autoindex
```

As previous but uses `hyperref`.

```
pdflatex sample-autoindex-hyp
makeglossaries sample-autoindex-hyp
pdflatex sample-autoindex-hyp
makeindex sample-autoindex-hyp
pdflatex sample-autoindex-hyp
```

Using \texttt{\textbackslash gls} within the value of the `name` key.

```
pdflatex sample-nested
makeglossaries sample-nested
pdflatex sample-nested
```
Enable entry-use counting (only index if used more than $n$ times, see §6.1).

```
pdflatex sample-entrycount
pdflatex sample-entrycount
makeglossaries sample-entrycount
pdflatex sample-entrycount
```

Enable use of per-unit entry-use counting (§6.1).

```
pdflatex sample-unitentrycount
pdflatex sample-unitentrycount
makeglossaries sample-unitentrycount
pdflatex sample-unitentrycount
```

Using the per-unit entry counting (§6.1) to only have one hyperlink per entry per page.

```
pdflatex sample-onelink
makeglossaries sample-onelink
pdflatex sample-onelink
```

Using link counting (§6.2) to only have one hyperlink per entry.

```
pdflatex sample-linkcount
makeglossaries sample-linkcount
pdflatex sample-linkcount
```

Insert “page” or “pages” before the location list.

```
pdflatex sample-pages
makeglossaries sample-pages
pdflatex sample-pages
pdflatex sample-pages
```

613
14. Supplementary Files

- `pdflatex sample-altmodifier`
- `makeglossaries sample-altmodifier`
- `pdflatex sample-altmodifier`
- `pdflatex sample-altmodifier`

Set the default options for commands like \gls and add an alternative modifier.

- `sample-mixedsort.tex`

- `pdflatex sample-mixedsort`
- `pdflatex sample-mixedsort`

Uses the optional argument of `\makeglossaries` to allow a mixture of `\printglossary` and `\printnoidxglossary`.

- `sample-external.tex`

- `pdflatex sample-external`
- `makeglossaries sample-external`
- `pdflatex sample-external`

Uses the `targeturl` attribute to allow for entries that should link to an external URL rather than to an internal glossary.

- `sample-fmt.tex`

- `pdflatex sample-fmt`
- `makeglossaries sample-fmt`
- `pdflatex sample-fmt`

Provides text-block commands associated with entries in order to use `\glsxtrfmt`.

- `sample-alias.tex`

- `pdflatex sample-alias`
- `makeglossaries sample-alias`
- `pdflatex sample-alias`

Uses the `alias` key (see §3.4).

- `sample-alttree.tex`
14. Supplementary Files

```
\pdflatex sample-alttree
\makeglossaries sample-alttree
\pdflatex sample-alttree
```

Uses the `glossaries-extra-stylemods` package with the `alttree` style (see §8.6.5).

```
\pdflatex sample-alttree-sym
\makeglossaries sample-alttree-sym
\pdflatex sample-alttree-sym
```

Another `alttree` example that measures the symbol widths.

```
\pdflatex sample-alttree-marginpar
\makeglossaries sample-alttree-marginpar
\pdflatex sample-alttree-marginpar
```

Another `alttree` example that puts the location list in the margin.

```
\pdflatex sample-onthefly
\makeglossaries sample-onthefly
\pdflatex sample-onthefly
```

Using on-the-fly commands. Terms with accents must have the `name` key explicitly set.

```
\xelatex sample-onthefly-xetex
\makeglossaries sample-onthefly-xetex
\xelatex sample-onthefly-xetex
```

Using on-the-fly commands with XƎLATEX. Terms with UTF-8 characters don’t need to have the `name` key explicitly set. Terms that contain commands must have the `name` key explicitly set with the commands removed from the label.

```
\xelatex sample-onthefly-utf8
```

```
14. Supplementary Files

Tries to emulate the previous sample file for use with \LaTeX through the starred version of \texttt{GlsXtrEnableOnTheFly}. This is a bit iffy and may not always work. Terms that contain commands must have the \texttt{name} key explicitly set with the commands removed from the label.

\texttt{sample-accsupp.tex}

Integrate \texttt{glossaries-accsupp}.

\texttt{sample-prefix.tex}

Integrate \texttt{glossaries-prefix}.

\texttt{sample-suppl-main.tex}

Uses \texttt{thevalue} to reference a location in the supplementary file \texttt{sample-suppl.tex}.

\texttt{sample-suppl-main-hyp.tex}
14. Supplementary Files

A more complicated version to the above that uses the hyperref package to reference a location in the supplementary file `sample-suppl-hyp.tex`. 
15. Multi-Lingual Support

There’s only one command provided by glossaries-extra that you’re likely to want to change in your document and that’s \abbreviationsname if you use the abbreviations package option to automatically create the glossary labelled abbreviations. If this command doesn’t already exist, it will be defined to “Abbreviations” if babel hasn’t been loaded, otherwise it will be defined as \acronymname (provided by glossaries), which is language-sensitive.

You can redefine \abbreviationsname in the usual way. For example:

```
\renewcommand*{\abbreviationsname}{List of Abbreviations}
```

Or using babel or polyglossia captions hook:

```
\appto\captionsenglish{%
    \renewcommand*{\abbreviationsname}{List of Abbreviations}%
}
```

Alternatively you can use the title key when you print the list of abbreviations. For example:

```
\printabbreviations[title={List of Abbreviations}]
```

or

```
\printabbreviations[typeabbreviations,title={List of Abbreviations}]
```

The other fixed text commands are the diagnostic messages, which shouldn’t appear in the final draft of your document.

The glossaries-extra package has the facility to load language modules (whose filename is in the form glossariesxtr-⟨language⟩.ldf) if they exist, but won’t warn if they don’t. If glossaries-extra-bib2gls is loaded via the record package option then the check for language resource files will additionally search for an associated language script file given by glossariesxtr-⟨script⟩.ldf where ⟨script⟩ is the four letter script identifier, such as Latn, associated with the given dialect. There’s no warning if the associated file isn’t found. The
script file is loaded after the dialect file.

If you want to write your own language module, you just need to create a file called `glossariesxtr-⟨lang⟩.ldf`, where `⟨lang⟩` identifies the language or dialect (see the `track-lang` package). For example, `glossariesxtr-french.1df`. The file should start with:

```
\ProvidesGlossariesExtraLang{⟨tag⟩}
```

The simplest code for this file is:

```
\ProvidesGlossariesExtraLang{french}[2015/12/09 v1.0]
\newcommand*{\glossariesxtrcaptionsfrench}{\%}
\def\abbreviationsname{Abr\'eviations}\%
\glossariesxtrcaptionsfrench
\ifsdef{captions\CurrentTrackedDialect}{}
\csappto{captions\CurrentTrackedDialect}{\%\%}
\glossariesxtrcaptionsfrench\%
\}%
\ifsdef{captions\CurrentTrackedLanguage}{}
\csappto{captions\CurrentTrackedLanguage}{\%\%}
\glossariesxtrcaptionsfrench\%
\}%
\}%
```

You can adapt this for other languages by replacing all instances of the language identifier `french` and the translated text `Abr\'eviations` as appropriate. You can also use the `ldf` file to provide rule blocks for a particular language for use with `bib2gls`’s custom sort rule. See §11.5 for further details.
15. Multi-Lingual Support

For further information on tracklang, see the tracklang documentation or Localisation with tracklang.tex.\footnote{dickimaw-books.com/latex/tracklang/}

This\.ldf file then needs to be put somewhere on \TeX’s path so that it can be found by glossaries-extra. You might also want to consider uploading it to CTAN\footnote{ctan.org/} so that it can be useful to others. (Please don’t send it to me. I already have more packages than I am able to maintain.)

If you additionally want to provide translations for the diagnostic messages used when a glossary is missing, you need to redefine the following commands:

\begin{verbatim}
\GlsXtrNoGlsWarningHead{⟨glossary-label⟩}{⟨file⟩}
\end{verbatim}

This produces the following text in English:

This document is incomplete. The external file associated with the glossary ‘⟨glossary-label⟩’ (which should be called ⟨file⟩) hasn’t been created.

\begin{verbatim}
\GlsXtrNoGlsWarningEmptyStart
\end{verbatim}

This produces the following text in English:

This has probably happened because there are no entries defined in this glossary.

\begin{verbatim}
\GlsXtrNoGlsWarningEmptyMain
\end{verbatim}

This produces the following text in English:

If you don’t want this glossary, add no\texttt{main} to your package option list when you load glossaries-extra.sty. For example:

\begin{verbatim}
\GlsXtrNoGlsWarningEmptyNotMain{⟨glossary-label⟩}
\end{verbatim}

This produces the following text in English:
Did you forget to use \texttt{type=⟨glossary-label⟩} when you defined your entries? If you tried to load entries into this glossary with \texttt{\textbackslash loadglsentries} did you remember to use \texttt{⟨⟨glossary-label⟩⟩} as the optional argument? If you did, check that the definitions in the file you loaded all had the type set to \texttt{\textbackslash glsdefaulttype}.

\begin{verbatim}
\GlsXtrNoGlsWarningCheckFile⟨⟨file⟩⟩
\end{verbatim}

This produces the following text in English:

Check the contents of the file \texttt{⟨⟨file⟩⟩}. If it’s empty, that means you haven’t indexed any of your entries in this glossary (using commands like \texttt{\textbackslash gls} or \texttt{\textbackslash glsadd}) so this list can’t be generated. If the file isn’t empty, the document build process hasn’t been completed.

\begin{verbatim}
\GlsXtrNoGlsWarningMisMatch
\end{verbatim}

This produces the following text in English:

You need to either replace \texttt{\textbackslash makenoidxglossaries} with \texttt{\textbackslash makeglossaries} or replace \texttt{\textbackslash printglossary} (or \texttt{\textbackslash printglossaries}) with \texttt{\textbackslash printnoidxglossary} (or \texttt{\textbackslash printnoidxglossaries}) and then rebuild this document.

\begin{verbatim}
\GlsXtrNoGlsWarningNoOut⟨⟨file⟩⟩
\end{verbatim}

This produces the following text in English:

The file \texttt{⟨⟨file⟩⟩} doesn’t exist. This most likely means you haven’t used \texttt{\textbackslash makeglossaries} or you have used \texttt{\textbackslash ofiles}. If this is just a draft version of the document, you can suppress this message using the \texttt{nomissingglstext} package option.

\begin{verbatim}
\GlsXtrNoGlsWarningTail
\end{verbatim}

This produces the following text in English:

This message will be removed once the problem has been fixed.

\begin{verbatim}
\GlsXtrNoGlsWarningBuildInfo
\end{verbatim}
This is advice on how to generate the glossary files.

\GlsXtrNoGlsWarningAutoMake{(glossary-label)}

This is the message produced when the \texttt{automake} option is used, but the document needs a rerun or the shell escape setting doesn’t permit the execution of the external application. This command also generates a warning in the transcript file.

See the documented code (glossaries-extra-code.pdf) for further details.
Part II.

Summaries and Index
Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>№</td>
<td>A counter is being described.</td>
</tr>
<tr>
<td>⏰</td>
<td>The syntax and usage of a command, environment or option etc.</td>
</tr>
<tr>
<td>⏰</td>
<td>A command, environment or option that is now deprecated.</td>
</tr>
<tr>
<td>⏰</td>
<td>A command, environment or option that should not be used with glossaries-extra.</td>
</tr>
<tr>
<td>❐</td>
<td>An important message.</td>
</tr>
<tr>
<td>❐</td>
<td>Prominent information.</td>
</tr>
<tr>
<td>⌨️</td>
<td>\LaTeX{} code to insert into your document.</td>
</tr>
<tr>
<td>📚</td>
<td>The definition of an option value.</td>
</tr>
<tr>
<td>📚</td>
<td>Problematic code which should be avoided.</td>
</tr>
<tr>
<td>📚</td>
<td>How the example code should appear in the PDF.</td>
</tr>
<tr>
<td>▼</td>
<td>An option that takes a value.</td>
</tr>
<tr>
<td>▼</td>
<td>A command-line application invocation that needs to be entered into a terminal or command prompt. See also “Incorporating makeglossaries or makeglossaries-lite or bib2gls into the document build&quot;.</td>
</tr>
<tr>
<td>☑️</td>
<td>A boolean option that is initially false.</td>
</tr>
<tr>
<td>☑️</td>
<td>A boolean option that is initially true.</td>
</tr>
<tr>
<td>📜</td>
<td>Text in a transcript or log file or written to STDOUT or STDERR.</td>
</tr>
<tr>
<td>📜</td>
<td>Code that requires a native Unicode engine (\TeX{} or Lua\TeX{}).</td>
</tr>
<tr>
<td>☑️</td>
<td>An option that doesn’t take a value.</td>
</tr>
<tr>
<td>⚠️</td>
<td>A warning.</td>
</tr>
</tbody>
</table>

---

\[\text{\textcopyright dickimaw-books.com/latex/buildglossaries}\]
Terms

American Standard Code for Information Interchange (ASCII)
A single-byte character encoding. Related blog article: Binary Files, Text Files and File Encodings.¹

Category post-description hook
A post-description hook called \glsxtrpostdesc\(\text{category}\) that is associated with a particular category.

Category post-link hook
A post-link hook called \glsxtrpostlink\(\text{category}\) that is associated with a particular category.

Category post-name hook
A post-name hook called \glsxtrpostname\(\text{category}\) that is associated with a particular category.

Command-line interface (CLI)
An application that doesn’t have a graphical user interface. That is, an application that doesn’t have any windows, buttons or menus and can be run in a command prompt or terminal.²

Display full form
The full form of an abbreviation obtained with the first use of \glsw{\textit{glsp1}} \(\text{glsp1}\) (or case-changing variants). This may or may not produce the same result as the inline full form the corresponding \glsxtrfull command, depending on the abbreviation style.

Entry location
The location of the entry in the document (obtained from the location counter or from the thevalue option). This defaults to the page number on which the entry has been referenced with any of the \glsw{\textit{gls}}-like, \glsw{\textit{glstext}}-like or \glsw{\textit{glsadd}} commands. An entry may have multiple locations that form a list.

Field
Entry data is stored in fields. These may have a corresponding key used to set the value, such as name or description, but field values may also be set using commands like \GlsXtrSetField, in which case there doesn’t need to be a corresponding key. Some fields are

¹ dickimaw-books.com/blog/binary-files-text-files-and-file-encodings/
² dickimaw-books.com/latex/novices/html/terminal.html
considered internal fields. If you are using \bib2gls, it will only recognise fields in the \bib file that have had a key defined in the document or that are special to \bib2gls.

**First use**
The first time an entry is used by a command that unsets the first use flag (or the first time since the flag was reset). For multi-entry sets, see first use.

**First use flag**
A conditional that keeps track of whether or not an entry has been referenced by any of the \gls-like commands (which can adjust their behaviour according to whether or not this flag is true). The conditional is true if the entry hasn’t been used by one of these commands (or if the flag has been reset) and false if it has been used (or if the flag has been unset). Note that multi-entries have their own flag that’s distinct from the first use flags of the individual elements.

**First use text**
The link text that is displayed on first use of the \gls-like commands.

**Group (letters, numbers, symbols)**
A logical division within a glossary that is typically a by-product of the indexing application’s sorting algorithm. See also Gallery: Logical Glossary Divisions (type vs group vs parent).

**Graphical user interface (GUI)**
An application that has windows, buttons or menus.

**Glossary**
Technically a glossary is an alphabetical list of words relating to a particular topic. For the purposes of describing the glossaries and glossaries-extra packages, a glossary is either the list produced by commands like \printglossary or \printunsrtglossary (which may or may not be ordered alphabetically) or a glossary is a set of entry labels where the set is identified by the glossary label or type.

**\gls-like**
Commands like \gls and \glsdisp that change the first use flag. These commands index the entry (if indexing is enabled), create a hyperlink to the entry’s glossary listing (if enabled) and unset the first use flag. These commands end with the post-link hook.

**\glstextr-like**
Commands like \glstextr and \glslink that don’t change the first use flag. These commands index the entry (if indexing is enabled) and create a hyperlink to the entry’s glossary listing (if enabled). These commands end with the post-link hook.

**Hierarchical level**
A number that indicates how many ancestors an entry has. An entry with no parent has hierarchical level 0. If an entry has a parent then the hierarchical level for the entry is one.

---

3dickimaw-books.com/gallery/index.php?label=logicaldivisions
more than the hierarchical level of the parent. Most styles will format an entry according
to its hierarchical level, giving prominence to level 0 entries, although some may have a
maximum supported limit. The "unsr" family of commands allow the hierarchical level to be
locally adjusted by adding an offset (leveloffset) or to have hierarchy ignored (flatten).

**Ignored glossary**

A glossary that has been defined using a command like \newignoredglossary. These
glossaries are omitted by iterative commands, such as \printglossaries and \printunsrtglossaries. An ignored glossary can only be displayed with \printunsrtglossary
or \printunsrtinnerglossary.

**Ignored location (or record)**

A location that uses glsignore as the encap. With bib2gls, this indicates that the entry
needs to be selected but the location isn’t added to the location list. With other methods, this
will simply create an invisible location, which can result in unwanted commas if the location
list has other items.

**Indexing application**

An application (piece of software) separate from \TeX that collates and sorts information
that has an associated page reference. Generally the information is an index entry but in this
case the information is a glossary entry. There are two main indexing applications that are
used with \TeX: makeindex and xindy. (There is also a new application called xindex, but
this isn’t supported by glossaries or glossaries-extra.) The glossaries-extra package addition-
ally supports bib2gls. These are all CLI applications.

**Indexing (or recording)**

The process of saving the entry location and any associated information that is required in
the glossary. In the case of makeindex and xindy, the entry location, encap, entry item
and sort value are written to a supplementary file associated with the glossary that is sub-
sequently read by makeindex/xindy. In the case of bib2gls and the "noidx" method, the
entry location, encap and label is written to the aux file. With record=nameref, the cur-
rent title and hyperlink target are also included. With bib2gls, each line of data in the aux
is referred to as a "record" and indexing is referred to as “recording”. The record=hybrid
option indexes twice: a bib2gls record in the aux file and a makeindex/xindy line in the
corresponding file, See §5.8.

**Inline full form**

The full form of an abbreviation obtained with \glsxtrfull or \glsxtrfullpl (or case-
changing variants). This may or may not produce the same result as the full form on first use
of the corresponding \gls-like command (the display full form), depending on the abbrevi-
ation style.

**Inner formatting**

The formatting applied by the innertextformat option, which redefines \glsxtrgenentry-
textfmt. The inner formatting can only be applied if \glsxtrgenentrytextfmt is embed-
ded within the entry’s display style.
Internal field
An internal field may refer to a key that shouldn’t be used in the bib file (internal field (bib2gls)), such as the group field, or it may refer to the label used to internally represent the field (which may or may not match the key used to set the field or may not have an associated key), such as useri which corresponds to the user1 key, or it may refer to a field that is only ever used internally that should not be explicitly modified, such as the field used to store the entry’s hierarchical level.

Internal field (bib2gls)
A field that is used or assigned by bib2gls that should typically not be used in the bib file.

Internal field label
The field label that forms part of the internal control sequence used to store the field value. This may or may not match the key used to assign the value when defining the entry. See the "Key to Field Mappings" table in the glossaries user manual.

Link text
The text produced by \gls-like and \glstext-like commands that have the potential to be a hyperlink.

Location counter
The counter used to obtain the entry location.

Location encap (format)
A command used to encapsulate an entry location. The control sequence name (without the leading backslash) is identified by the format key. The default encap is \glsnumberformat.

Location list
A list of entry locations (also called a number list). May be suppressed for all glossaries with the package option nonumberlist or for individual glossaries with nonumberlist. With bib2gls, the list may also be suppressed with save-locations=false.

Multi-entry first use
The first time a multi-entry is referenced (or the first time since the multi-entry first use flag was reset). This is not necessary the first use of any of the individual entries that form the set.

Multi-entry first use flag
A conditional that keeps track of whether or not a multi-entry has been used. This is distinct from the first use flags of the individual elements.

Multi-entry subsequent use
When a multi-entry that has already been marked as used is referenced. This is not necessary the subsequent use of any of the individual entries that form the set.
Post-description hook

A hook (`\glspostdescription`) included in some glossary styles that is used after the description is displayed. The glossaries-extra package modifies this command to provide additional hooks, including category post-description hooks. The glossaries-extra-stylemods package modifies the predefined styles provided with glossaries to ensure that they all use `\glspostdescription` to allow the category post-description hooks to be implemented.

Post-link hook

A hook (command) that is used after link text to allow code to be automatically added. The base glossaries package provides a general purpose hook `\glspostlinkhook`. The glossaries-extra package modifies this command to provide additional hooks, including category post-link hooks.

Post-name hook

A hook (command) that is used after the name is displayed in glossary styles. These hooks are implemented by `\glossentryname`, which needs to be present in the glossary style. The main hook is `\glsxtrpostnamehook`, which implements auto-indexing (see §12), performs a general purpose hook `\glsxtrapostnamehook` and a category-specific hook `\glsxtrpostname(category)`.

Print “unsrt” glossary commands (and environment)

The set of commands used for displaying a glossary or partial glossary that have “unsrt” in the name: `\printunsrtglossary`, `\printunsrtglossaries` (which internally uses `\printunsrtglossary`), and `\printunsrtinnerglossary`. These all simply iterate over the list of entries associated with the given glossary, in the order in which they were added to the glossary (hence “unsrt”, which is short for “unsorted”). The way that bib2gls works is that it sorts the entries according to the resource options and adds the entries to the glossary in the required order. These commands may be used with or without bib2gls. If you don’t use bib2gls, you will need to manually ensure that the entries are added in the desired order. The `printunsrtglossarywrap` environment may also be included in this category, although it only sets up the start and end of the glossary for use with `\printunsrtinnerglossary`.

Resource file

The glstex file created by bib2gls and loaded by `\GlsXtrLoadResources` (or `\glsxtrresourcefile`).

Resource set

All the settings (resource options) and entries associated with a particular instance of `\GlsXtrLoadResources` (or `\glsxtrresourcefile`).

Standalone entry

Normally the link text target is created in the glossary, where the entry’s name and description (and optionally other information, such as the symbol) are shown. It may be that you don’t want a list but need to have the name and description somewhere in the document text.
Terms

These are standalone entries. Whilst it is possible to simply use \glsentryname{} and \glsentrydesc{}, the glossaries-extra package provides a way inserting the name that includes the hypertarget for the link text and obeys the post-name hook.

Subsequent use
Using an entry that unsets the first use flag when it has already been unset.

Unicode Transformation Format (8-bit) (UTF-8)
A variable-width encoding that uses 8-bit code units. This means that some characters are represented by more than one byte. \LaTeX{} and \LuaLaTeX{} treat the multi-byte sequence as a single token, but the older \TeX{} formats have single-byte tokens, which can cause complications, although these have mostly been addressed with the newer kernels introduced over the past few years. Related blog article: Binary Files, Text Files and File Encodings.

Whatsit
A command whose execution is delayed or an OS-specific special command. This includes writing to external files (which is what indexing does).

\footnotesize\begin{footnote}{\url{dickimaw-books.com/blog/binary-files-text-files-and-file-encodings/}}
\end{footnote}
Glossary Entry Keys Summary

These are options that can be passed to commands that define entries, such as `\newglossary-entry` or `\newabbreviation`.

\begin{verbatim}
access={⟨text⟩} \text{ (requires accsupp)}
\end{verbatim}

Accessibility text corresponding to the `name` field. This field will be automatically set by `\newabbreviation`, if not provided and the `nameshortaccess` attribute is set. See §9.1.

\begin{verbatim}
alias={⟨xr-label⟩} \text{ glossaries-extra}
\end{verbatim}

Behaves in a similar manner to `see={\[seealso⟨name⟩\]}⟨xr-label⟩` but also sets up aliasing which makes the link text hyperlink to ⟨xr-label⟩ instead.

\begin{verbatim}
category={⟨category-label⟩} \text{ initial: general glossaries-extra}
\end{verbatim}

The entry’s category (must be a simple label).

\begin{verbatim}
counter={⟨counter-name⟩}
\end{verbatim}

If set, the value indicates the location counter to use by default when indexing this entry (overrides the counter associated with the glossary or the `counter` package option).

\begin{verbatim}
description={⟨text⟩}
\end{verbatim}

The entry’s description, as displayed in the glossary. If required in the text, use `\glsdesc` (if indexing and hyperlinks are required) or `\glsentrydesc`. Glossary styles should use `\glossentrydesc` and `\glspostdescription` to incorporate the post-description hook.
Glossary Entry Keys Summary

\[\text{descriptionaccess}=\{(\text{text})\} \quad \text{(requires accsupp)}\]

Accessibility text corresponding to the \text{description} field.

\[\text{descriptionplural}=\{(\text{text})\} \quad \text{\(\)}\]

The plural form of the entry’s description, if applicable. If omitted, this is set to the same value as the \text{description}, since descriptions tend not to be a singular entity.

\[\text{descriptionpluralaccess}=\{(\text{text})\} \quad \text{(requires accsupp)}\]

Accessibility text corresponding to the \text{descriptionplural} field.

\[\text{first}=\{(\text{first})\} \quad \text{\(\)}\]

The entry’s text, as displayed on first use of \texttt{\textbackslash gls}-like commands. Note that using an abbreviation style or post-link hooks is a more flexible approach. If omitted, this value is assumed to be the same as the \text{text} key.

\[\text{firstaccess}=\{(\text{text})\} \quad \text{(requires accsupp)}\]

Accessibility text corresponding to the \text{first} field. This field will be automatically set by \texttt{\textbackslash newabbreviation}, if not provided and the \text{firstshortaccess} attribute is set. See §9.1.

\[\text{firstplural}=\{(\text{text})\} \quad \text{\(\)}\]

The entry’s plural form, as displayed on first use of plural \texttt{\textbackslash gls}-like commands, such as \texttt{\textbackslash glsp1}. If this key is omitted, then the value will either be the same as the \text{plural} field, if the \text{first} key wasn’t used, or the value will be taken from the \text{first} key with \texttt{\textbackslash glsplural-} suffix appended.

\[\text{firstpluralaccess}=\{(\text{text})\} \quad \text{(requires accsupp)}\]
Glossary Entry Keys Summary

Accessibility text corresponding to the firstplural field. This field will be automatically set by `\newabbreviation`, if not provided and the firstshortaccess attribute is set. See §9.1.

\begin{verbatim}
\verb|group|\{\langle\text{group-label}\rangle\}\phantomsection\glsresetall
\end{verbatim}

§3.2; 34

The group label that identifies which group the entry belongs to. This key is only available with the record=only and record=nameref options, and is set by bib2gls, if invoked with --group or -g. This is an internal key assigned by bib2gls as a by-product of sorting. Explicit use without reference to the order of entries can result in fragmented groups. The corresponding title can be set with `\glsxtrsetgrouptitle`, although this is more commonly done implicitly within the glstex file.

\begin{verbatim}
\verb|location|\{\langle\text{location-list}\rangle\}\phantomsection\glsresetall
\end{verbatim}

§3.2; 35

The formatted location list used by the “unsrt” family of commands. This key is only available with the record option and is set by bib2gls unless save-locationsfalse is set.

\begin{verbatim}
\verb|long|\{\langle\text{long-form}\rangle\}\phantomsection\glsresetall
\end{verbatim}

A field that is set by `\newabbreviation` to the entry’s long (unabbreviated) form. It typically shouldn’t be used explicitly with `\newglossaryentry` as `\newabbreviation` makes other modifications to ensure that when the entry is referenced with the \gls-like commands, it will obey the appropriate abbreviation style. If you are using bib2gls then this field should be used in the bib file when defining abbreviations.

\begin{verbatim}
\verb|longaccess|\{\langle\text{text}\rangle\}\phantomsection\glsresetall (requires accsupp)
\end{verbatim}

Accessibility text corresponding to the long field.

\begin{verbatim}
\verb|longplural|\{\langle\text{long-form}\rangle\}\phantomsection\glsresetall
\end{verbatim}

As long but the plural form.
Glossary Entry Keys Summary

longpluralaccess=⟨(text)⟩  
Accessibility text corresponding to the longplural field.

name=⟨(text)⟩
The entry’s name, as displayed in the glossary. This typically isn’t used outside of the glossary (the text and plural keys are used instead). However, if there is a need to specifically display the entry name, use \glsname{} (if indexing and hyperlinks are required) or \glsentryname{}. Glossary styles should use \glossentryname{}, which uses \glsentryname{} and incorporates the post-name hooks and related attributes.

parent=⟨parent-label⟩
The label of the entry’s parent (from which the entry’s hierarchical level is obtained).

plural=⟨(text)⟩
The entry’s plural form, as displayed on subsequent use of plural \gls-like commands, such as \glspl{}. This should be the appropriate plural form of the value provided by the text key. If omitted, this value is assumed to be the value of the text key with \glspl{}{} appended.

pluralaccess=⟨(text)⟩  
Accessibility text corresponding to the plural field. This field will be automatically set by \newabbreviation{}, if not provided and the textshortaccess attribute is set. See §9.1.

prefix=⟨(text)⟩  
glossaries-prefix v3.14a+
The subsequent use singular prefix.
Glossary Entry Keys Summary

prefixfirst=\{\langle text \rangle\}  \gls@glossaries-prefix v3.14a+

The first use singular prefix.

prefixfirstplural=\{\langle text \rangle\}  \gls@glossaries-prefix v3.14a+

The first use plural prefix.

prefixplural=\{\langle text \rangle\}  \gls@glossaries-prefix v3.14a+

The subsequent use plural prefix.

see=\{[[\langle tag \rangle]]\langle xr-list \rangle\}

With the base glossaries package this simply triggers an automatic cross-reference with \glssee. The glossaries-extra package additionally saves the value. Use \texttt{autooseeindex=false} to prevent the automatic cross-reference. The \langle tag \rangle defaults to \texttt{\seename} and \langle xr-list \rangle should be a comma-separated list of entries that have already been defined.

seealso=\{\langle xr-list \rangle\}  \gls@glossaries-extra

Behaves in a similar manner to see=\{[\langle seealso\rangle\langle xr-list \rangle\text{.}

short=\{\langle short-form \rangle\}

A field that is set by \texttt{\newabbreviation} (and \texttt{\newacronym}) to the entry’s short (abbreviated) form. It typically shouldn’t be used explicitly with \texttt{\newglossaryentry} as \texttt{\newabbreviation} makes other modifications to ensure that when the entry is referenced with the \texttt{\gls-like} commands, it will obey the appropriate abbreviation style. If you are using \texttt{bib2gls} then this field should be used in the \texttt{bib} file when defining abbreviations.

shortaccess=\{\langle text \rangle\}  \gls (requires accsupp)

635
Glossary Entry Keys Summary

Accessibility text corresponding to the `short` field. This field will be automatically set by `\newabbreviation`, if not provided. See §9.1.

\[\text{shortplural}={\langle\text{short-form}\rangle}\]

As `short` but the plural form. The default is obtained by appending the abbreviation plural suffix, but this behaviour can be altered by category attributes. See §4 for further details.

\[\text{shortpluralaccess}={\langle\text{text}\rangle}\] (requires `accsupp`)

Accessibility text corresponding to the `shortplural` field. This field will be automatically set by `\newabbreviation`, if not provided. See §9.1.

\[\text{sort}={\langle\text{value}\rangle}\quad\text{initial:}\langle\text{entry name}\rangle\]

Specifies the value to use for sorting (overrides the default). This key is usually required for `xindy` if the `name` key only contains commands (for example, the entry is a symbol), but explicitly using this key in other contexts can break certain sort methods. Don’t use the `sort` field with `bib2gls`.

\[\text{symbol}={\langle\text{symbol}\rangle}\quad\text{initial:}\langle\text{\relax}\rangle\]

The entry’s associated symbol (optional), which can be displayed with `\glsymbol` (if indexing and hyperlinks are required) or with `\glsentrysymbol`.

\[\text{symbolaccess}={\langle\text{text}\rangle}\] (requires `accsupp`)

Accessibility text corresponding to the `symbol` field.

\[\text{symbolplural}={\langle\text{symbol plural}\rangle}\]

\[^{1}\text{dickimaw-books.com/gallery/index.php?label=bib2gls-sorting}\]
The plural form of the \texttt{symbol}, if applicable, which can be displayed with \texttt{\glsymbolplural} (if indexing and hyperlinks are required) or with \texttt{\glsentrysymbolplural}. If omitted, this value is set to the same as the \texttt{symbol} key (since symbols usually don’t have a plural form).

\begin{verbatim}
symbolpluralaccess={⟨text⟩}
\end{verbatim}

Accessibility text corresponding to the \texttt{symbolplural} field.

\begin{verbatim}
text={⟨text⟩}
\end{verbatim}

The entry’s text, as displayed on subsequent use of \texttt{\gls}-like commands. If omitted, this value is assumed to be the same as the \texttt{name} key.

\begin{verbatim}
textaccess={⟨text⟩}
\end{verbatim}

Accessibility text corresponding to the \texttt{text} field. This field will be automatically set by \texttt{\newabbreviation}, if not provided and the \texttt{textshortaccess} attribute is set. See §9.1.

\begin{verbatim}
type=(glossary-label)
\end{verbatim}

Assigns the entry to the glossary identified by \texttt{(glossary-label)}.

\begin{verbatim}
user1={⟨text⟩}
\end{verbatim}

A generic field, which can be displayed with \texttt{\glsuseri} (if indexing and hyperlinks are required) or with \texttt{\glsentryuseri}.

\begin{verbatim}
user1access={⟨text⟩}
\end{verbatim}

Accessibility text corresponding to the \texttt{user1} field.
### Glossary Entry Keys Summary

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Generic Field Description</th>
<th>Accessibility Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>user2</td>
<td>A generic field, which can be displayed with \glsuserii (if indexing and hyperlinks are required) or with \glsentryuserii.</td>
<td>Accessibility text corresponding to the user2 field.</td>
</tr>
<tr>
<td>user3</td>
<td>A generic field, which can be displayed with \glsuseriii (if indexing and hyperlinks are required) or with \glsentryuseriii.</td>
<td>Accessibility text corresponding to the user3 field.</td>
</tr>
<tr>
<td>user4</td>
<td>A generic field, which can be displayed with \glsuseriv (if indexing and hyperlinks are required) or with \glsentryuseriv.</td>
<td>Accessibility text corresponding to the user4 field.</td>
</tr>
<tr>
<td>user5</td>
<td>A generic field, which can be displayed with \glsuserv (if indexing and hyperlinks are required) or with \glsentryuserv.</td>
<td></td>
</tr>
</tbody>
</table>
Glossary Entry Keys Summary

user5access=\{(text)\}

Accessibility text corresponding to the user5 field.

user6=\{(text)\}

A generic field, which can be displayed with \glsuservi (if indexing and hyperlinks are required) or with \glsentryuservi.

user6access=\{(text)\}

Accessibility text corresponding to the user6 field.
Glossary Entry Fields Summary

These are internal fields that don’t have a corresponding key.

\begin{verbatim}
childcount=\{\langle number\rangle\}
\end{verbatim}

Used with the `save-child-count` resource option to store the entry’s child count.

\begin{verbatim}
childlist=\{\langle entry-label-list\rangle\}
\end{verbatim}

Used with the `save-child-count` resource option to store the entry’s children as an etoolbox internal list.

\begin{verbatim}
indexcounter=\{\langle target-name\rangle\}
\end{verbatim}

Used with the `indexcounter` package option and the `save-index-counter` resource option. The value is set to the hyperlink target of the first `wrglossary` location or the first instance for a specific location encap.

\begin{verbatim}
locist=\{\langle etoolbox list\rangle\}
\end{verbatim}

Used by \verb|\printnoidxglossary| to provide the locations. The value is an etoolbox list of individual locations which are obtained from the aux file. This field will also be used by the “unsrt” family of commands if `location` isn’t set.

\begin{verbatim}
recordcount=\{\langle number\rangle\}
\end{verbatim}

Used with the `--record-count` switch to store the total number of records for the associated entry.
**Glossary Entry Fields Summary**

recordcount.\langle counter\rangle=\langle number\rangle

Used with the --record-count switch to store the total number of records with the location counter \langle counter\rangle for the associated entry.

recordcount.\langle counter\rangle.\langle location\rangle=\langle number\rangle

Used with the --record-count-unit switch to store the total number of records with the location counter \langle counter\rangle set to \langle location\rangle for the associated entry.

record.\langle counter\rangle=\langle location\rangle

Used with \texttt{\LaTeX}XtrRecordCounter to store an etoolbox internal list of locations (without encap) corresponding to the given counter.

secondarygroup=\langle group-label\rangle

Used by \texttt{bib2gls} to store the group label obtained from the secondary sort.

useri

Corresponds to user1 key.

userii

Corresponds to user2 key.

useriii

Corresponds to user3 key.
\texttt{\textbackslash gls-Like and \textbackslash glstext-Like Options}

Summary

Most (but not all) of these options can be used in the optional argument of all the \texttt{\gls-like}, \texttt{\glstext-like} and \texttt{\glsadd} commands.

\begin{itemize}
\item \texttt{counter=⟨counter-name⟩}
\end{itemize}

The location counter.

\begin{itemize}
\item \texttt{format=⟨cs-name⟩}
\end{itemize}

The control sequence name (without the leading backslash) that should be used to encapsulate the entry location.

\begin{itemize}
\item \texttt{hyper=⟨boolean⟩} \hspace{1cm} default: true; initial: true
\end{itemize}

Determines whether or not the link text should have a hyperlink (provided hyperlinks are supported).

\begin{itemize}
\item \texttt{hyperoutside=⟨boolean⟩} \hspace{1cm} default: true; initial: true \hspace{1cm} glossaries-extra v1.21+[§5.1.2; 190]
\end{itemize}

Determines whether the hyperlink should be inside or outside of \texttt{\glstextformat}.

\begin{itemize}
\item \texttt{innertextformat=⟨csname⟩} \hspace{1cm} initial: glsxtrdefaultentrytextfmt \hspace{1cm} glossaries-extra v1.49+[§5.1.2; 191]
\end{itemize}

The name of the control sequence to use for the inner formatting.
### \gls{}-Like and \glistext{}-Like Options Summary

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Initial</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>local</code></td>
<td>boolean</td>
<td>true; false</td>
<td>$\odot$ glossaries v3.04+</td>
</tr>
<tr>
<td>If true use <code>\glslocalunset</code> to unset the first use flag, otherwise use <code>\glsunset</code> (only applies to <code>\gls</code>-like commands).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>noindex</code></td>
<td>boolean</td>
<td>true; false</td>
<td>$\odot$ glossaries-extra v1.30+</td>
</tr>
<tr>
<td>If true this option will suppress indexing. If you are using <code>bib2gls</code>, you may want to consider using <code>format=glsignore</code> to prevent a location but ensure that the entry is selected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>postunset</code></td>
<td>value</td>
<td>global; global</td>
<td>$\odot$ glossaries-extra v1.49+</td>
</tr>
<tr>
<td>Determines whether or not to unset the first use flag after the link text. The value may be one of: global, local or none (only applies to <code>\gls</code>-like commands).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>prefix</code></td>
<td>link-prefix</td>
<td></td>
<td>$\Xi$ glossaries-extra v1.31+</td>
</tr>
<tr>
<td>The prefix to use for the entry’s hyperlink target.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>prerest</code></td>
<td>value</td>
<td>local; none</td>
<td>$\Xi$ glossaries-extra v1.49+</td>
</tr>
<tr>
<td>Determines whether or not to reset the entry before the link text. Allowed values: none (no reset), local (localise the reset) and global.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>preunset</code></td>
<td>value</td>
<td>local; none</td>
<td>$\Xi$ glossaries-extra v1.49+</td>
</tr>
<tr>
<td>Determines whether or not to unset the entry before the link text. Allowed values: none (no unset), local (localise the unset) and global.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>textformat</code></td>
<td>csname</td>
<td></td>
<td>$\Xi$ glossaries-extra v1.30+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
643
\gls{-Like and \glstext{-Like Options Summary}

The name of the control sequence to use instead of \glstextformat to encapsulate the link text.

`theHvalue={the-H-value}`

Set the hyper location to this value instead of obtaining it from `\theH(counter)`.

`thevalue={location}`

Set the location to this value instead of obtaining it from the location counter.

`types={\{glossary list\}}`

Only available with `\glsaddall`, the value is the list of glossaries to iterate over.

`wrgloss={position} initial: before`

Determines whether to do the indexing before or after the link text. Allowed values: before and after.
## Multi-Entry Set Options Summary

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all=(option list)</code></td>
<td>Options to pass to the \gls-lik command for each element.</td>
</tr>
<tr>
<td><code>category=(category-label)</code></td>
<td>The category to assign to the multi-entry set.</td>
</tr>
<tr>
<td><code>encapmain=(value)</code></td>
<td>The value to pass to the <code>format</code> option for the main entry.</td>
</tr>
<tr>
<td><code>encapothers=(value)</code></td>
<td>The value to pass to the <code>format</code> option for the “other” elements.</td>
</tr>
<tr>
<td><code>firstprefix=(value)</code></td>
<td>The prefix to use on first use of the multi-entry.</td>
</tr>
<tr>
<td><code>firstskipmain=(boolean)</code></td>
<td>Determines whether or not to skip the main entry on first use.</td>
</tr>
</tbody>
</table>

- `§7.10; 363`
- `§7.9.6; 362`  
- `§7.9.2; 358`  
- `§7.9.2; 358`  
- `§7.9.4; 360`  
- `§7.9.5; 361`
Multi-Entry Set Options Summary

firstskipothers=⟨boolean⟩  
\textit{default: true; initial: false}  
§7.9.5; 361

Determines whether or not to skip the “other” elements on first use.

firstsuffix=⟨value⟩  
§7.9.4; 360

The suffix to use on first use of the multi-entry.

hyper=⟨boolean⟩  
\textit{default: true}  
§7.10; 364

Indicates whether or not to use hyperlinks, if supported, for all elements. This option is for use in the optional argument of \texttt{\textbackslash gls} and can be set implicitly with the default behaviour of the * and + modifiers.

hyper=⟨value⟩  
\textit{initial: individual}  
§7.9.6; 361

Indicates which elements should have hyperlinks, if supported. This option is a multi-entry setting, see §7.9.

indexmain=⟨value⟩  
\textit{default: true; initial: true}  
§7.9.1; 358

Indicates if the main element should be indexed, should only be indexed on first use or should not indexed.

indexothers=⟨value⟩  
\textit{default: true; initial: true}  
§7.9.1; 358

Indicates if the “other” elements should be indexed, should only be indexed on first use or should not indexed.

main=⟨option list⟩  
§7.10; 363

Options to pass to the \texttt{\textbackslash gls}-like command for the main entry.
**Multi-Entry Set Options Summary**

- **mglsopts=⟨option list⟩**
  The default options to pass to commands like \mglso.

- **mpostlink=⟨value⟩**
  Indicates whether or not the multi-entry post-link hook should be enabled and, if so, whether it should only be enabled on first or subsequent use.

- **mpostlinkelement=⟨value⟩**
  Indicates which post-link hook to use if the multi-entry post-link hook has been enabled.

- **multiunset=⟨value⟩**
  Indicates whether or not the multi-entry first use flag should be unset.

- **others=⟨option list⟩**
  Options to pass to the \gls-like command for the “other” elements.

- **postlinks=⟨value⟩**
  Indicates which post-link hooks should be enabled.

- **presetlocal=⟨boolean⟩**
  Indicates whether or not the prereset options should have a local or global effect.

- **resetall=⟨boolean⟩**
  Indicates whether or not the prereset options should have a local or global effect.
Multi-Entry Set Options Summary

Indicates whether or not to reset all elements’ first use flag before using \glsls.

\texttt{resetmain=⟨boolean⟩} \hspace{1cm} \textit{default: true; initial: false} \hspace{1cm} \textsuperscript{§7.10; 364}

Indicates whether or not to reset the main entry’s first use flag before using \glsls.

\texttt{resetothers=⟨boolean⟩} \hspace{1cm} \textit{default: true; initial: false} \hspace{1cm} \textsuperscript{§7.10; 364}

Indicates whether or not to reset all “other” elements’ first use flag before using \glsls.

\texttt{setup=⟨option list⟩} \hspace{1cm} \textsuperscript{§7.10; 363}

Multi-entry options that will override any conflicting options already assigned to the multi-entry.

\texttt{textformat=⟨value⟩} \hspace{1cm} \textit{initial: @firstofone} \hspace{1cm} \textsuperscript{§7.9.6; 362}

The control sequence name of the command that should encapsulate the entire content.

\texttt{unsetall=⟨boolean⟩} \hspace{1cm} \textit{default: true; initial: false} \hspace{1cm} \textsuperscript{§7.10; 365}

Indicates whether or not to unset all elements’ first use flag before using \glsls.

\texttt{unsetmain=⟨boolean⟩} \hspace{1cm} \textit{default: true; initial: false} \hspace{1cm} \textsuperscript{§7.10; 365}

Indicates whether or not to unset the main entry’s first use flag before using \glsls.

\texttt{unsetothers=⟨boolean⟩} \hspace{1cm} \textit{default: true; initial: false} \hspace{1cm} \textsuperscript{§7.10; 365}

Indicates whether or not to unset all “other” elements’ first use flag before using \glsls.
### Multi-Entry Set Options Summary

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default</th>
<th>Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>usedprefix</code></td>
<td>The prefix to use on subsequent use of the multi-entry.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>usedskipmain</code></td>
<td>Determines whether or not to skip the main entry on subsequent use.</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td><code>usedskipothers</code></td>
<td>Determines whether or not to skip the “other” elements on subsequent use.</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td><code>usedsuffix</code></td>
<td>The suffix to use on subsequent use of the multi-entry.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

The prefix to use on subsequent use of the multi-entry.

Determines whether or not to skip the main entry on subsequent use.

Determines whether or not to skip the “other” elements on subsequent use.

The suffix to use on subsequent use of the multi-entry.
Print [Unsrt|noidx] Glossary Options

Summary

Most (but not all) of these options can be used in the optional argument of all the print glossary commands: \printglossary, \printnoidxglossary, \printunsrtglossary and \printunsrtinnerglossary. Some may be used in the optional argument of the printunsrtglossarywrap environment.

entrycounter=\langle boolean\rangle  \hfill default: \text{true}; \ initial: \text{false} \square \text{glossaries v4.08+}

If true, enable the entry counter.

flatten=\langle boolean\rangle  \hfill default: \text{true}; \ initial: \text{false} \square \text{glossaries-extra v1.49+}

§8.3; 383

If true, treats all entries as though they have the same hierarchical level (the value of leveloffset). This option is only available for the “unsrt” family of commands and the printunsrtglossarywrap environment.

groups=\langle boolean\rangle  \hfill default: \text{true}; \ initial: \text{true} \square \text{glossaries-extra v1.44+}

§8.3; 383

Enables group formation. This option is only available for the “unsrt” family of commands and the printunsrtglossarywrap environment. Note that no groups will be formed when invoking bib2gls with the default --no-group, regardless of this setting.

label=\langle label\rangle  \quad \Psi \text{glossaries-extra v1.39+}

§8.3; 382

Adds \textbackslash label\{\langle label\rangle\} to the start of the glossary (after the title). Not available with \textbackslash printunsrtinnerglossary.

leveloffset=\langle offset\rangle  \hfill initial: \text{0} \Psi \text{glossaries-extra v1.44+}

§8.3; 382
Set or increment the hierarchical level offset. If \( \langle \text{offset} \rangle \) starts with ++ then the current offset is incremented by the given amount otherwise the current offset is set to \( \langle \text{offset} \rangle \). For example, an entry with a normal hierarchical level of 1 will be treated as though it has hierarchical level \( 1 + \langle \text{offset} \rangle \). This option is only available for the “unsrt” family of commands and the printunsrtglossarywrap environment.

\[
\text{nogroupskip}=\langle \text{boolean} \rangle \quad \text{default: true; initial: false} \]

Suppress the gap implemented by some glossary styles between groups.

\[
\text{nonumberlist}=\langle \text{boolean} \rangle \quad \text{default: true; initial: false} \]

Suppress the location list. Note that nonumberlist=false will have no effect with the save-locations=false resource option as there won’t be any location lists to display.

\[
\text{nopostdot}=\langle \text{boolean} \rangle \quad \text{default: true; initial: false} \]

Suppress the post-description punctuation.

\[
\text{numberedsection}=\langle \text{value} \rangle \quad \text{default: nolabel; initial: false} \]

Indicates whether or not glossary section headers will be numbered and also if they should automatically be labelled. The numberedsection package option will change the default setting to match.

\[
\text{postamble}=\langle \text{text} \rangle \quad \text{glossaries-extra v1.49+} \]

Redefines \glossarypostamble to \( \langle \text{text} \rangle \).

\[
\text{preamble}=\langle \text{text} \rangle \quad \text{glossaries-extra v1.49+} \]

Redefines \glossarypreamble to \( \langle \text{text} \rangle \).
prefix=⟨prefix⟩

Redefines \glolinkprefix to ⟨prefix⟩.

sort=⟨method⟩

Only available with \printnoidxglossary, this indicates how the glossary should be ordered.

style=⟨style-name⟩

Use the ⟨style-name⟩ glossary style.

subentrycounter=⟨boolean⟩  default: true; initial: false  glossaries v4.08+

If true, enable the sub-entry counter.

target=⟨boolean⟩  default: true; initial: true  glossaries-extra v1.12+

If true, each entry in the glossary should have a hypertarget created, if supported by the glossary style and if hyperlinks are enabled.

targetnameprefix=⟨prefix⟩  glossaries-extra v1.20+

Inserts ⟨prefix⟩ at the start of the hypertarget names.

title=⟨text⟩

Sets the glossary title (overriding the default).
Sets the glossary toc title (overriding the default).

Identifies the glossary to display.
Abbreviation Styles Summary

Abbreviations defined using \newabbreviation will follow the style associated with the entry’s category. If there is no style associated with the entry’s category, the style for the abbreviation category is used (the default is long-short). Note that glossaries-extra re-defines \newacronym to use \newabbreviation with category=acronym so any entry defined with \newacronym will use the abbreviation style for the acronym category (the default is short-nolong).

footnote-desc

alias: short-footnote-desc

footnote-em

alias: short-em-footnote

footnote-sc

alias: short-sc-footnote

footnote-sm

alias: short-sm-footnote

footnote

alias: short-footnote

long-desc-em

alias: long-noshort-em-desc

long-desc-sc

alias: long-noshort-sc-desc
Abbreviation Styles Summary

- **long-desc-sm**
  alias: `long-noshort-sm-desc`

- **long-desc**
  alias: `long-noshort-desc`

- **long-em-noshort-em-desc-noreg**
  An abbreviation style like `long-em-noshort-em-desc` but sets the `regular` attribute to false.

- **long-em-noshort-em-desc**
  An abbreviation style like `long-noshort-desc` but formats both the long and short form in an emphasized font (`\textit{}`).

- **long-em-noshort-em-noreg**
  An abbreviation style like `long-em-noshort-em` but sets the `regular` attribute to false.

- **long-em-noshort-em**
  An abbreviation style like `long-noshort` but formats both the long and short form in an emphasized font (`\textit{}`).

- **long-em-short-em-desc**
  An abbreviation style like `long-short-desc` but formats both the long and short form in an emphasized font (`\textit{}`).
### Abbreviation Styles Summary

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>long-em-short-em</strong></td>
<td>An abbreviation style like <code>long-short</code> but formats both the long and short form in an emphasized font (\textit{}).</td>
</tr>
<tr>
<td><strong>long-em</strong></td>
<td>alias: <code>long-noshort-em</code></td>
</tr>
<tr>
<td><strong>long-hyphen-noshort-desc-noreg</strong></td>
<td>An abbreviation style like <code>long-noshort-desc</code> but checks if the inserted material starts with a hyphen (use with markwords attribute).</td>
</tr>
<tr>
<td><strong>long-hyphen-noshort-noreg</strong></td>
<td>An abbreviation style like <code>long-hyphen-short-hyphen</code> but doesn’t show the short form on first use.</td>
</tr>
<tr>
<td><strong>long-hyphen-postshort-hyphen-desc</strong></td>
<td>An abbreviation style like <code>long-hyphen-short-hyphen-desc</code> but places the insert and parenthetical material in the post-link hook.</td>
</tr>
<tr>
<td><strong>long-hyphen-postshort-hyphen</strong></td>
<td>An abbreviation style like <code>long-hyphen-short-hyphen</code> but places the insert and parenthetical material in the post-link hook.</td>
</tr>
<tr>
<td><strong>long-hyphen-short-hyphen-desc</strong></td>
<td>An abbreviation style like <code>long-hyphen-short-hyphen</code> but the description must be supplied.</td>
</tr>
</tbody>
</table>
**Abbreviation Styles Summary**

- **long-hyphen-short-hyphen**
  An abbreviation style like `long-short` but checks if the inserted material starts with a hyphen (use with `markwords` or `markshortwords` attributes).

- **long-noshort-desc-noreg**
  As `long-noshort-desc` but it will set the `regular` attribute to `false`.

- **long-noshort-desc**
  An abbreviation style that only shows the long form on first use and subsequent use. The short form won’t be showed unless you use a command like `\glsxtrshort`. The `description` key must be supplied. The full form will only be showed with commands like `\glsxtrfull`. This style sets the `regular` attribute to `true`.

- **long-noshort-em-desc**
  An abbreviation style like `long-noshort-desc` but formats the short form in an emphasized font (`\emph`).

- **long-noshort-em**
  An abbreviation style like `long-noshort` but formats the short form in an emphasized font (`\emph`).

- **long-noshort-noreg**
  As `long-noshort` but it will set the `regular` attribute to `false`.

- **long-noshort-sc-desc**

657
An abbreviation style like \texttt{long-noshort-desc} but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

\texttt{long-noshort-sc}  
75

An abbreviation style like \texttt{long-noshort} but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

\texttt{long-noshort-sm-desc}  
77

An abbreviation style like \texttt{long-noshort-desc} but formats the short form in a smaller font (\textsmaller). The \texttt{relsize} package is must be loaded.

\texttt{long-noshort-sm}  
76

An abbreviation style like \texttt{long-noshort} but formats the short form in a smaller font (\textsmaller). The \texttt{relsize} package is must be loaded.

\texttt{long-noshort}  
74, 75

An abbreviation style that only shows the long form on first use and subsequent use. The short form won't be showed unless you use a command like \texttt{glsxtrshort}. The full form will only be shown with commands like \texttt{glsxtrfull}. This style sets the \texttt{regular} attribute to true.

\texttt{long-only-short-only-desc}  
112

An abbreviation style like \texttt{long-only-short-only} but the description must be supplied.

\texttt{long-only-short-only}  
111

An abbreviation style that only shows the long form on first use and only shows the short form on subsequent use.
Abbreviation Styles Summary

- **long-only-short-sc-only-desc**
  - An abbreviation style like `long-only-short-sc-only` but the description must be supplied.

- **long-only-short-sc-only**
  - An abbreviation style like `long-only-short-only` but uses small caps for the short form.

- **long-postshort-sc-user-desc**
  - An abbreviation style like `long-postshort-sc-user` but the description must be supplied.

- **long-postshort-sc-user**
  - An abbreviation style like `long-postshort-user` but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

- **long-postshort-user-desc**
  - An abbreviation style like `long-postshort-user` but the description must be supplied.

- **long-postshort-user**
  - An abbreviation style like `long-short-user` but the parenthetical content is placed in the post-link hook.

- **long-sc alias: long-noshort-sc**

- **long-short-desc**
Abbreviation Styles Summary

As `long-short` but the description must be supplied in \(\text{\textlangle options\rangle}\).

\begin{itemize}
  \item `long-short-em-desc`
  \item An abbreviation style like `long-short-desc` but formats the short form in an emphasized font \(\textit{\textlangle emph\rangle}\).
  \item `long-short-em`
  \item An abbreviation style like `long-short` but formats the short form in an emphasized font \(\textit{\textlangle emph\rangle}\).
  \item `long-short-sc-desc`
  \item An abbreviation style like `long-short-desc` but formats the short form in a small caps font \(\textsc{}\). The short form should therefore be in lowercase.
  \item `long-short-sc`
  \item An abbreviation style like `long-short` but formats the short form in a small caps font \(\textsc{}\). The short form should therefore be in lowercase.
  \item `long-short-sm-desc`
  \item An abbreviation style like `long-short-desc` but formats the short form in a smaller font \(\texttextsmaller{}\). The \texttt{relsize} package is must be loaded.
  \item `long-short-sm`
  \item An abbreviation style like `long-short` but formats the short form in a smaller font \(\texttextsmaller{}\). The \texttt{relsize} package is must be loaded.
\end{itemize}
Abbreviation Styles Summary

**long-short-user-desc**

An abbreviation style like `long-short-user` but the description must be supplied.

**long-short-user**

An abbreviation style like `long-short` but includes the value of the field identified by `\gls-xtruserfield` (if set) in the parenthetical content.

**long-short**

An abbreviation style that shows the long form followed by the short form on first use. If the `<insert>` argument is used with the `\gls-like` or `\glstext-like` commands, it will be placed after the long form on first use. On subsequent use, only the short form is shown (followed by `<insert>`, if provided). This style sets the `regular` attribute to `false` (which means that the `\gls-like` commands won’t use the `first/firstplural` or `text/plural` values).

**long-sm** *alias: long-noshort-sm*

**long** *alias: long-noshort*

**nolong-short-em**

An abbreviation style like `nolong-short` but formats the short form in an emphasized font (`\emph`).

**nolong-short-noreg**

As `nolong-short` but it will set the `regular` attribute to `false`. 
**Abbreviation Styles Summary**

**nolong-short-sc**

An abbreviation style like `nolong-short` but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

**nolong-short-sm**

An abbreviation style like `nolong-short` but formats the short form in a smaller font (\textsmaller). The \relsize package is must be loaded.

**nolong-short**

As `short-nolong` but the inline full form shows the long form followed by the short form in parentheses.

**postfootnote-desc**  
*alias: short-postfootnote-desc*

**postfootnote-em**  
*alias: short-em-postfootnote*

**postfootnote-sc**  
*alias: short-sc-postfootnote*

**postfootnote-sm**  
*alias: short-sm-postfootnote*

**postfootnote**  
*alias: short-postfootnote*
An abbreviation style like `short-footnote-desc` but formats the short form in an emphasized font (`\textit`).

An abbreviation style like `short-footnote` but formats the short form in an emphasized font (`\textit`).

An abbreviation style like `short-long-desc` but formats the short form in an emphasized font (`\textit`).

An abbreviation style like `short-long-desc` but formats both the long and short form in an emphasized font (`\textit`).

An abbreviation style like `short-long` but formats both the long and short form in an emphasized font (`\textit`).
Abbreviation Styles Summary

An abbreviation style like `short-long` but formats the short form in an emphasized font (\emph).

\begin{itemize}
  \item \texttt{short-em-nolong-desc} \hfill 72, 73
\end{itemize}

An abbreviation style like `short-nolong-desc` but formats the short form in an emphasized font (\emph).

\begin{itemize}
  \item \texttt{short-em-nolong} \hfill 71, 72
\end{itemize}

An abbreviation style like `short-nolong` but formats the short form in an emphasized font (\emph).

\begin{itemize}
  \item \texttt{short-em-postfootnote-desc} \hfill 129
\end{itemize}

An abbreviation style like `short-postfootnote-desc` but formats the short form in an emphasized font (\emph).

\begin{itemize}
  \item \texttt{short-em-postfootnote} \hfill 128
\end{itemize}

An abbreviation style like `short-postfootnote` but formats the short form in an emphasized font (\emph).

\begin{itemize}
  \item \texttt{short-em} \hfill 72 \hfill alias: \texttt{short-em-nolong}
\end{itemize}

\begin{itemize}
  \item \texttt{short-footnote-desc} \hfill 115, 116
\end{itemize}

As `short-footnote` but the description must be supplied in \langle options\rangle.

\begin{itemize}
  \item \texttt{short-footnote} \hfill 114, 115
\end{itemize}
Abbreviation Styles Summary

An abbreviation style that shows the short form with the long form as a footnote on first use. If the \texttt{⟨insert⟩} argument is used with the \texttt{\gls}-like or \texttt{\glstext}-like commands, it will be placed after the short form, before the footnote marker, on first use. On subsequent use, only the short form is shown (followed by \texttt{⟨insert⟩}, if provided). The inline full form shows the short form followed by the long form in parentheses. This style sets the \texttt{regular} attribute to \texttt{false} (which means that the \texttt{\gls}-like commands won’t use the \texttt{first/first-plural} or \texttt{text/plural} values). This style also sets the \texttt{nohyperfirst} attribute to \texttt{true} to avoid nesting the footnote marker link. If you want hyperlinks on first use, use the \texttt{short-postfootnote} style instead.

\begin{itemize}
  \item \texttt{short-hyphen-long-hyphen-desc} \hfill 110
  \item \texttt{short-hyphen-long-hyphen} \hfill 108
  \item \texttt{short-hyphen-postlong-hyphen-desc} \hfill 110
  \item \texttt{short-hyphen-postlong-hyphen} \hfill 109
  \item \texttt{short-long-desc} \hfill 94
\end{itemize}

An abbreviation style like \texttt{short-hyphen-long-hyphen} but the description must be supplied.

An abbreviation style like \texttt{short-long} but checks if the inserted material starts with a hyphen (use with \texttt{markwords} or \texttt{markshortwords} attributes).

An abbreviation style like \texttt{short-hyphen-long-hyphen-desc} but the insert and parenthetical material are placed in the post-link hook.

An abbreviation style like \texttt{short-hyphen-long-hyphen} but the insert and parenthetical material are placed in the post-link hook.

As \texttt{short-long} but the \texttt{description} must be supplied in \texttt{⟨options⟩}. 

665
**Abbreviation Styles Summary**

- **short-long-user-desc**
  
  An abbreviation style like `short-long-user` but the description must be supplied.

- **short-long-user**
  
  An abbreviation style like `short-long` but includes the value of the field identified by `\gls-xtruserfield` (if set) in the parenthetical content.

- **short-long**
  
  An abbreviation style that shows the short form followed by the long form on first use. If the `<insert>` argument is used with the `\gls-like` or `\glstext-like` commands, it will be placed after the short form on first use. On subsequent use, only the short form is shown (followed by `<insert>`, if provided). This style sets the `regular` attribute to false (which means that the `\gls-like` commands won’t use the `first/firstplural` or `text/plural` values).

- **short-nolong-desc-noreg**
  
  As `short-nolong-desc` but it will set the `regular` attribute to false.

- **short-nolong-desc**
  
  As `short-nolong` but the description must be supplied in `<options>`.

- **short-nolong-noreg**
  
  As `short-nolong` but it will set the `regular` attribute to false.

- **short-nolong**
  
  An abbreviation style that only shows the short form on first use and subsequent use. The long form won’t be showed unless you use a command like `\glsxtrlong`. The full form will
only be shown with commands like \glsxtrfull. This style sets the regular attribute to true.

**short-postfootnote-desc**

As short-postfootnote but the description must be supplied in ⟨options⟩.

**short-postfootnote**

Similar to short-footnote but the footnote is placed in the post-link hook.

**short-postlong-user-desc**

An abbreviation style like short-postlong-user but the description must be supplied.

**short-postlong-user**

An abbreviation style like short-long but includes the value of the field identified by \glsxtruserfield (if set) in the parenthetical content, which is placed in the post-link hook.

**short-sc-desc** alias: short-sc-nolong-desc

An abbreviation style like short-footnote-desc but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

**short-sc-footnote-desc**

An abbreviation style like short-footnote but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.
Abbreviation Styles Summary

**short-sc-long-desc**

An abbreviation style like `short-long-desc` but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

**short-sc-long**

An abbreviation style like `short-long` but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

**short-sc-nolong-desc**

An abbreviation style like `short-nolong-desc` but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

**short-sc-nolong**

An abbreviation style like `short-nolong` but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

**short-sc-postfootnote-desc**

An abbreviation style like `short-postfootnote-desc` but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

**short-sc-postfootnote**

An abbreviation style like `short-postfootnote` but formats the short form in a small caps font (\textsc). The short form should therefore be in lowercase.

**short-sc**

*alias: short-sc-nolong*
<table>
<thead>
<tr>
<th>Abbreviation Style</th>
<th>Alias: Abbreviation Style</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>short-sm-desc</strong></td>
<td><strong>short-sm-nolong-desc</strong></td>
</tr>
<tr>
<td><strong>short-sm-footnote-desc</strong></td>
<td></td>
</tr>
<tr>
<td>An abbreviation style like short-footnote-desc but formats the short form in a smaller font (\textsmaller). The relsize package is must be loaded.</td>
<td></td>
</tr>
<tr>
<td><strong>short-sm-footnote</strong></td>
<td></td>
</tr>
<tr>
<td>An abbreviation style like short-footnote but formats the short form in a smaller font (\textsmaller). The relsize package is must be loaded.</td>
<td></td>
</tr>
<tr>
<td><strong>short-sm-long-desc</strong></td>
<td></td>
</tr>
<tr>
<td>An abbreviation style like short-long-desc but formats the short form in a smaller font (\textsmaller). The relsize package is must be loaded.</td>
<td></td>
</tr>
<tr>
<td><strong>short-sm-long</strong></td>
<td></td>
</tr>
<tr>
<td>An abbreviation style like short-long but formats the short form in a smaller font (\textsmaller). The relsize package is must be loaded.</td>
<td></td>
</tr>
<tr>
<td><strong>short-sm-nolong-desc</strong></td>
<td></td>
</tr>
<tr>
<td>An abbreviation style like short-nolong-desc but formats the short form in a smaller font (\textsmaller). The relsize package is must be loaded.</td>
<td></td>
</tr>
<tr>
<td><strong>short-sm-nolong</strong></td>
<td></td>
</tr>
<tr>
<td>An abbreviation style like short-nolong but formats the short form in a smaller font (\textsmaller). The relsize package is must be loaded.</td>
<td></td>
</tr>
</tbody>
</table>
Abbreviation Styles Summary

**short-sm-postfootnote-desc**

An abbreviation style like `short-postfootnote-desc` but formats the short form in a smaller font (\textsmaller). The relsize package is must be loaded.

**short-sm-postfootnote**

An abbreviation style like `short-postfootnote` but formats the short form in a smaller font (\textsmaller). The relsize package is must be loaded.

**short-sm** \textit{alias: short-sm-nolong}

**short** \textit{alias: short-nolong}
### Glossary Styles Summary

The default style may be set with \setglossarystyle or use:

\usepackage[stylemods=⟨name⟩,style=⟨style-name⟩]{glossaries-extra}

where the style is provided by package glossary-⟨name⟩. The default style can be overridden for individual glossaries with the style option. For a summary of all available styles, see Gallery: Predefined Styles.¹

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abbr-long-short</td>
<td>This style displays the glossary in a table for (either longtable or tabular) with two columns: the long form and the short form.</td>
</tr>
<tr>
<td>abbr-short-long</td>
<td>This style displays the glossary in a table for (either longtable or tabular) with two columns: the short form and the long form.</td>
</tr>
<tr>
<td>altlist</td>
<td>As list but starts the description on a new line.</td>
</tr>
<tr>
<td>alttree</td>
<td>Like tree but the width of the widest name must be supplied (using a command like \glssetwidest).</td>
</tr>
<tr>
<td>alttreegroup</td>
<td></td>
</tr>
</tbody>
</table>

¹dickimaw-books.com/gallery/index.php?label=glossaries-styles

§8.7.2.6; 466
As alttree but has headers at the start of each group.

| bookindex                  | glossary-bookindex |

This style is designed for indexes. Symbols and descriptions are not shown. Since descriptions aren’t shown, there’s no post-description hook.

| index                      | glossary-tree      |

A hierarchical style that supports up to level 2, similar to normal indexes, but symbols and descriptions are shown.

| indexgroup                 | glossary-tree      |

As index but has headers at the start of each group.

| inline                     | glossary-inline    |

A compact style with all entries listed in the same paragraph and no groups, locations or symbols.

| list                       | glossary-list      |

This style uses the description environment and places the entry name in the optional argument of \item. Symbols and sub-entry names are not shown.

| listdotted                 | glossary-list      |

A list-like style that has a dotted leader between the name and description. The location list isn’t shown.

| listgroup                  | glossary-list      |
As list but has headers at the start of each group.

```
listhypergroup    glossary-list
```

As listgroup but has a row at the start with hyperlinks to each group.

```
long-booktabs     glossary-longbooktabs v4.21+
```

This style displays the glossary using `longtable` and horizontal rules from the `booktabs` package.

```
long-custom1-name glossary-longextra v1.50+
```

This style displays the glossary in a table for (either `longtable` or `tabular`) with two columns: the custom 1 field and the name.

```
long-custom2-name glossary-longextra v1.50+
```

This style displays the glossary in a table for (either `longtable` or `tabular`) with three columns: the custom 1 field, custom 2 field, and the name.

```
long-custom3-name glossary-longextra v1.50+
```

This style displays the glossary in a table for (either `longtable` or `tabular`) with four columns: the custom 1 field, custom 2 field, custom 3 field, and the name.

```
long-desc-custom1-name glossary-longextra v1.50+
```

This style displays the glossary in a table for (either `longtable` or `tabular`) with three columns: the description, the custom 1 field and the name.

```
long-desc-custom2-name glossary-longextra v1.50+
```
This style displays the glossary in a table for (either longtable or tabular) with four columns: the description, the custom 1 field, the custom 2 field and the name.

<table>
<thead>
<tr>
<th>long-desc-custom3-name</th>
<th>glossary-longextra v1.50+</th>
</tr>
</thead>
</table>

This style displays the glossary in a table for (either longtable or tabular) with five columns: the description, the custom 1 field, the custom 2 field, the custom 3 field and the name.

<table>
<thead>
<tr>
<th>long-desc-name</th>
<th>glossary-longextra v1.37+</th>
</tr>
</thead>
</table>

This style displays the glossary in a table for (either longtable or tabular) with two columns: the description and the name.

<table>
<thead>
<tr>
<th>long-desc-sym-name</th>
<th>glossary-longextra v1.37+</th>
</tr>
</thead>
</table>

This style displays the glossary in a table for (either longtable or tabular) with three columns: the description, symbol and name.

<table>
<thead>
<tr>
<th>long-desc-sym</th>
<th>glossary-longextra v1.49+</th>
</tr>
</thead>
</table>

This style displays the glossary in a table for (either longtable or tabular) with two columns: the description and the symbol.

<table>
<thead>
<tr>
<th>long-loc-desc-name</th>
<th>glossary-longextra v1.37+</th>
</tr>
</thead>
</table>

This style displays the glossary in a table for (either longtable or tabular) with three columns: the location list, description and name.

<table>
<thead>
<tr>
<th>long-loc-desc-sym-name</th>
<th>glossary-longextra v1.37+</th>
</tr>
</thead>
</table>

This style displays the glossary in a table for (either longtable or tabular) with four columns: the location list, description, symbol and name.

<table>
<thead>
<tr>
<th>long-loc-desc-sym-name</th>
<th>glossary-longextra v1.37+</th>
</tr>
</thead>
</table>

674
### Glossary Styles Summary

<table>
<thead>
<tr>
<th>Style Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long-loc-sym-desc-name</td>
<td>Displays the glossary in a table for (either longtable or tabular) with four columns: the location list, symbol, description and name.</td>
</tr>
<tr>
<td>long-name-custom1</td>
<td>Displays the glossary in a table for (either longtable or tabular) with two columns: the name and the custom 1 field.</td>
</tr>
<tr>
<td>long-name-custom1-desc</td>
<td>Displays the glossary in a table for (either longtable or tabular) with three columns: the name, the custom 1 field and the description.</td>
</tr>
<tr>
<td>long-name-custom2</td>
<td>Displays the glossary in a table for (either longtable or tabular) with three columns: the name, custom 1 field and custom 2 field.</td>
</tr>
<tr>
<td>long-name-custom2-desc</td>
<td>Displays the glossary in a table for (either longtable or tabular) with four columns: the name, the custom 1 field, the custom 2 field and the description.</td>
</tr>
<tr>
<td>long-name-custom3</td>
<td>Displays the glossary in a table for (either longtable or tabular) with four columns: the name, custom 1 field, custom 2 field and custom 3 field.</td>
</tr>
<tr>
<td>long-name-custom3-desc</td>
<td>Displays the glossary in a table for (either longtable or tabular) with four columns: the name, custom 1 field, custom 2 field and custom 3 field.</td>
</tr>
</tbody>
</table>
This style displays the glossary in a table for (either longtable or tabular) with five columns: the name, the custom 1 field, the custom 2 field, the custom 3 field and the description.

\texttt{long-name-desc-loc} \quad \textit{glossary-longextra v1.37+} \quad \S8.7.2.3; 458

This style displays the glossary in a table for (either longtable or tabular) with three columns: the name, description and location list.

\texttt{long-name-desc-sym-loc} \quad \textit{glossary-longextra v1.37+} \quad \S8.7.2.4; 459

This style displays the glossary in a table for (either longtable or tabular) with four columns: the name, description, symbol and location list.

\texttt{long-name-desc-sym} \quad \textit{glossary-longextra v1.37+} \quad \S8.7.2.2; 456

This style displays the glossary in a table for (either longtable or tabular) with three columns: the name, description and symbol.

\texttt{long-name-desc} \quad \textit{glossary-longextra v1.37+} \quad \S8.7.2.1; 455

This style displays the glossary in a table for (either longtable or tabular) with two columns: the name and the description.

\texttt{long-name-sym-desc-loc} \quad \textit{glossary-longextra v1.37+} \quad \S8.7.2.4; 460

This style displays the glossary in a table for (either longtable or tabular) with four columns: the name, symbol, description and location list.

\texttt{long-name-sym-desc} \quad \textit{glossary-longextra v1.37+} \quad \S8.7.2.2; 456

This style displays the glossary in a table for (either longtable or tabular) with three columns: the name, symbol and description.
**Glossary Styles Summary**

<table>
<thead>
<tr>
<th>Style</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>long-sym-desc-name</code></td>
<td>glossary-longextra v1.37+</td>
</tr>
<tr>
<td><code>long-sym-desc</code></td>
<td>glossary-longextra v1.49+</td>
</tr>
<tr>
<td><code>long</code></td>
<td>glossary-long</td>
</tr>
<tr>
<td><code>longheader</code></td>
<td>glossary-long</td>
</tr>
<tr>
<td><code>longragged</code></td>
<td>glossary-longragged</td>
</tr>
<tr>
<td><code>mcolindex</code></td>
<td>glossary-mcols</td>
</tr>
<tr>
<td><code>mcolindexgroup</code></td>
<td>glossary-mcols</td>
</tr>
</tbody>
</table>

This style displays the glossary in a table for (either `longtable` or `tabular`) with three columns: the symbol, description and name.

This style displays the glossary in a table for (either `longtable` or `tabular`) with two columns: the symbol and the description.

This style uses the `longtable` environment (provided by the `longtable` package). Symbols and sub-entry names are not shown.

This style uses the `longtable` environment (provided by the `longtable` package) with a header row. Symbols and sub-entry names are not shown.

This style displays the glossary using `longtable` with ragged right paragraph formatting for the description column.

As index but puts the content inside a `multicols` environment.

As `mcolindex` but has headers at the start of each group.
Glossary Styles Summary

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcoltree</td>
<td>As tree but puts the content inside a multicols environment.</td>
</tr>
<tr>
<td>super</td>
<td>This style displays the glossary using supertabular.</td>
</tr>
<tr>
<td>superragged</td>
<td>This style displays the glossary using supertabular with ragged right paragraph formatting for the description column.</td>
</tr>
<tr>
<td>table</td>
<td>This style is specific to \printunsrttable.</td>
</tr>
<tr>
<td>topic</td>
<td>This style is designed for hierarchical glossaries where the top-level entry represents a topic.</td>
</tr>
<tr>
<td>topicmcols</td>
<td>Similar to topic but the sub-entries are placed in a multicols environment.</td>
</tr>
<tr>
<td>tree</td>
<td>A hierarchical style that supports unlimited levels (although a deep hierarchy may not fit the available line width) with that shows symbols and descriptions.</td>
</tr>
</tbody>
</table>

678
Glossary Styles Summary

As tree but has headers at the start of each group.

| treehypergroup | glossary-tree |

As treegroup but has a row at the start with hyperlinks to each group.

| treenoname | glossary-tree |

Like tree but the child entries don’t have their name shown.
Command Summary

\glsxtr@doglossary
glossaries-extra v1.08+

Internal command used within the construction of the glossary code by the “unsrt” family of commands. Should not be used or modified but \printunsrtglossarypredoglossary can be defined to show the definition of this command for debugging purposes.

\glsxtr@multientry{(options)}{(multi-label)}{(main-label)}{(list)}
glossaries-extra v1.48+

Information in the aux about a multi-label defined in the previous \LaTeX run.

\glsxtr@starttoc{(toc)}

Set to the definition of \starttoc when glossaries-extra loads.

\glsxtr@markboth{(left text)}{(right text)}

Set to the definition of \markboth when glossaries-extra loads.

\glsxtr@markright{(text)}

Set to the definition of \markright when glossaries-extra loads.
### Command Summary

Redefines \glsxtrifinmark to do its first argument (*true*).

\@glsxtrnotinmark
glossaries-extra v1.07+

§5.3.3; 227

Redefines \glsxtrifinmark to do its first argument (*true*).

\[ \text{A synonym for } \textbackslash cGLS \text{ defined by the shortcuts=abbreviations package option.} \]

\[ \Ab\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle inset\rangle\} \quad \text{modifiers: } * + \langle alt-mod \rangle \]

§4.3.2; Table 4.1

\[ \text{A synonym for } \textbackslash cGls \text{ defined by the shortcuts=abbreviations package option.} \]

\[ \ab\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle inset\rangle\} \quad \text{modifiers: } * + \langle alt-mod \rangle \]

§4.3.2; Table 4.1

\[ \text{A synonym for } \textbackslash cgls \text{ defined by the shortcuts=abbreviations package option.} \]

\[ \text{Expands to the title of the abbreviations glossary. The default is “Abbreviations” or } \textbackslash \text{acronym-name } \text{if } \text{babel} \text{ has been detected.} \]

\[ \text{Style-sensitive abbreviation suffix. This is the command that’s actually used in the value of the shortplural key when an entry is defined with } \textbackslash \text{newabbreviation} \text{ (unless suppressed with the noshortplural attribute). This command is redefined by the abbreviation styles} \]

\[ \text{initial: } \textbackslash glsxtrabbrvpluralsuffix \]

§2.1; 9

<table>
<thead>
<tr>
<th>Abbreviationsname</th>
<th>initial: Abbreviations</th>
<th>glossaries-extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>\abbreviationsname</td>
<td>(language-sensitive)</td>
<td></td>
</tr>
</tbody>
</table>

\abbrvpluralsuffix

\[ \text{initial: } \textbackslash glsxtrabbrvpluralsuffix \]

171

681
Command Summary

to \glsxtrabbrvpluralsuffix or the style’s custom suffix command (such as \glsxtrscsuffix).

\ABP{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩  \(\S4.3.2;\) Table 4.1

A synonym for \cGLSp1 defined by the shortcuts=abbreviations package option.

\Abp{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩  \(\S4.3.2;\) Table 4.1

A synonym for \cGlspl defined by the shortcuts=abbreviations package option.

\abp{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩  \(\S4.3.2;\) Table 4.1

A synonym for \cGlspl defined by the shortcuts=abbreviations package option.

\AC{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩  \(\S4.3.2;\) Table 4.1

A synonym for \cGLS defined by the shortcuts=ac package option.

\Ac{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩  \(\S4.3.2;\) Table 4.1

A synonym for \cGLS or \Gls defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\ac{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩  \(\S4.3.2;\) Table 4.1

A synonym for \cGLs or \Gls defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\ACF{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩  \(\S4.3.2;\) Table 4.1

682
A synonym for \GLSxtrfull defined by the shortcuts=ac package option.

```
\Acf\{\langle\text{options}\rangle\}\{\langle\text{entry-label}\rangle\}\{\langle\text{inset}\rangle\} \hspace{1em} \text{modifiers: } * + \langle\text{alt-mod}\rangle
```

§4.3.2; Table 4.1

A synonym for \Glsxtrfull or \Acrfull defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

```
\acf\{\langle\text{options}\rangle\}\{\langle\text{entry-label}\rangle\}\{\langle\text{inset}\rangle\} \hspace{1em} \text{modifiers: } * + \langle\text{alt-mod}\rangle
```

§4.3.2; Table 4.1

A synonym for \Glsxtrfullpl or \Acrfullpl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

```
\ACFP\{\langle\text{options}\rangle\}\{\langle\text{entry-label}\rangle\}\{\langle\text{inset}\rangle\} \hspace{1em} \text{modifiers: } * + \langle\text{alt-mod}\rangle
```

§4.3.2; Table 4.1

A synonym for \GLSxtrfullpl defined by the shortcuts=ac package option.

```
\Acfp\{\langle\text{options}\rangle\}\{\langle\text{entry-label}\rangle\}\{\langle\text{inset}\rangle\} \hspace{1em} \text{modifiers: } * + \langle\text{alt-mod}\rangle
```

§4.3.2; Table 4.1

A synonym for \Glsxtrfullpl or \Acrfullpl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

```
\acfp\{\langle\text{options}\rangle\}\{\langle\text{entry-label}\rangle\}\{\langle\text{inset}\rangle\} \hspace{1em} \text{modifiers: } * + \langle\text{alt-mod}\rangle
```

§4.3.2; Table 4.1

A synonym for \Glsxtrlong defined by the shortcuts=ac package option.

```
\Acl\{\langle\text{options}\rangle\}\{\langle\text{entry-label}\rangle\}\{\langle\text{inset}\rangle\} \hspace{1em} \text{modifiers: } * + \langle\text{alt-mod}\rangle
```

§4.3.2; Table 4.1
A synonym for \Glsxtrlong or \Acrlong defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\[\texttt{\acl{\langle options\rangle}{\langle entry-label\rangle}{\langle inset\rangle}} \quad \text{modifiers: } * + \langle alt-mod\rangle\]

A synonym for \gllsxtrlongpl or \acrlongpl defined by the shortcuts=ac package option.

\[\texttt{\Aclp{\langle options\rangle}{\langle entry-label\rangle}{\langle inset\rangle}} \quad \text{modifiers: } * + \langle alt-mod\rangle\]

A synonym for \GLSxtrlongpl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.

\[\texttt{\ACLp{\langle options\rangle}{\langle entry-label\rangle}{\langle inset\rangle}} \quad \text{modifiers: } * + \langle alt-mod\rangle\]

A synonym for \cGLSpl defined by the shortcuts=ac package option.

\[\texttt{\Acp{\langle options\rangle}{\langle entry-label\rangle}{\langle inset\rangle}} \quad \text{modifiers: } * + \langle alt-mod\rangle\]

A synonym for \cGlspl or \glspl defined by the shortcuts=ac or shortcuts=acronyms package option, respectively.
A synonym for \cglslpl or \glspl defined by the \texttt{shortcuts=ac} or \texttt{shortcuts=acronyms} package option, respectively.

As \acrfull but sentence case.

Displays the full form of an acronym. Only for use with the base glossaries package’s acronym mechanism. This command is not compatible with \texttt{\newabbreviation}.

As \acrfullpl but sentence case.

Displays the plural full form of an acronym. Only for use with the base glossaries package’s acronym mechanism. This command is not compatible with \texttt{\newabbreviation}.

As \acrlong but sentence case.

Displays the long form of an acronym. Only for use with the base glossaries package’s acronym mechanism. This command is not compatible with \texttt{\newabbreviation}. 
Command Summary

\acrlongpl[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩}  modifiers: * + ⟨alt-mod⟩

As \acr but sentence case.

\acrlongpl[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩}  modifiers: * + ⟨alt-mod⟩

Displays the plural long form of an acronym. Only for use with the base glossaries package’s acronym mechanism. This command is not compatible with \newabbreviation.

\acronymfont{⟨text⟩}; glossaries v1.0+

Used to encapsulate the acronym short form on subsequent use by the base glossaries package. This is redefined by glossaries-extra to use \glsabbrvfont.

\acroynname  (language-sensitive)

Expands to the title of the acronym glossary.

\acronymtype  \glsdefaulttype; glossaries

Expands to the label of the default acronym glossary. The \acr or \acronym package option will redefine this to \acr. The abbreviations package option will redefine this to \glsxtrabbrvtype if \acronym/\acr isn’t used.

\acrpluralsuffix

Used for the plural suffixes for the base package’s acronym mechanism. Not used with glossaries-extra’s abbreviations, which use \glsxtrabbrvpluralsuffix, \abbrvplural-suffix and commands provided for use with particular abbreviation styles. This command should not be used with glossaries-extra.
Command Summary

\texttt{\textbackslash Acrshort\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle insert\rangle\}} \quad \text{modifiers: } * + \langle \text{alt-mod} \rangle

As \texttt{\textbackslash acrshort} but sentence case.

\texttt{\textbackslash Acrshort\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle insert\rangle\}} \quad \text{modifiers: } * + \langle \text{alt-mod} \rangle

Displays the short form of an acronym. Only for use with the base glossaries package’s acronym mechanism. This command is not compatible with \texttt{\textbackslash newabbreviation}.

\texttt{\textbackslash Acrshortpl\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle insert\rangle\}} \quad \text{modifiers: } * + \langle \text{alt-mod} \rangle

As \texttt{\textbackslash acrshort} but sentence case.

\texttt{\textbackslash Acrshortpl\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle insert\rangle\}} \quad \text{modifiers: } * + \langle \text{alt-mod} \rangle

Displays the plural short form of an acronym. Only for use with the base glossaries package’s acronym mechanism. This command is not compatible with \texttt{\textbackslash newabbreviation}.

\texttt{\textbackslash ACS\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle inset\rangle\}} \quad \text{modifiers: } * + \langle \text{alt-mod} \rangle

A synonym for \texttt{\textbackslash GLSxtrshort} defined by the \texttt{shortcuts=ac} package option.

\texttt{\textbackslash Ac\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle inset\rangle\}} \quad \text{modifiers: } * + \langle \text{alt-mod} \rangle

A synonym for \texttt{\textbackslash Glsxtrshort} or \texttt{\textbackslash Acrshort} defined by the \texttt{shortcuts=ac} or \texttt{shortcuts=acronyms} package option, respectively.

\texttt{\textbackslash acs\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle inset\rangle\}} \quad \text{modifiers: } * + \langle \text{alt-mod} \rangle

A synonym for \texttt{\textbackslash Glsxtrshort} or \texttt{\textbackslash acrshort} defined by the \texttt{shortcuts=ac} or \texttt{shortcuts=acronyms} package option, respectively.

687
A synonym for \GLSxtrshortpl defined by the \texttt{shortcuts=ac} package option.

A synonym for \Glsxtrshortpl or \Acrshortpl defined by the \texttt{shortcuts=ac} or \texttt{shortcuts=acronyms} package option, respectively.

A synonym for \Glsxtrshortpl defined by the \texttt{shortcuts=acronyms} package option, respectively.

A synonym for \Glsxtrfull defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \Glsxtrfull defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \Glsxtrfull defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \GLSxtrfullpl defined by the \texttt{shortcuts=abbreviations} package option.
A synonym for \texttt{\glsxtrfullpl} defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \texttt{\glsxtrfullpl} defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \texttt{\GLSxtrlong} defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \texttt{\GLSxtrlong} defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \texttt{\glsxtrlong} defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \texttt{\glsxtrlong} defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \texttt{\GLSxtrlongpl} defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \texttt{\GLSxtrlongpl} defined by the \texttt{shortcuts=abbreviations} package option.

A synonym for \texttt{\glsxtrlongpl} defined by the \texttt{shortcuts=abbreviations} package option.
Command Summary

A synonym for \glsxtrlongpl defined by the shortcuts=abbreviations package option.

\Alpha  

Defined with \providecommand, this just does \mathrm{A}.

\alsoname  initial: see also  (language-sensitive)

Used as a cross-reference tag (provided by language packages, such as babel).

\andname  initial: \&

Used by \glsseelastsep (provided by glossaries if not already defined).

\apptoglossarypreamble[⟨type⟩]{⟨text⟩}  

Appends (locally) ⟨text⟩ to the preamble for the glossary identified by ⟨type⟩. If ⟨type⟩ is omitted, \glsdefaulttype is assumed.

\AS{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩

A synonym for \GLSxtrshort defined by the shortcuts=abbreviations package option.

\As{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩

A synonym for \Glsxtrshort defined by the shortcuts=abbreviations package option.

\as{⟨options⟩}{⟨entry-label⟩}{⟨inset⟩}  modifiers: * + ⟨alt-mod⟩

A synonym for \glsxtrshort defined by the shortcuts=abbreviations package option.
Command Summary

\ASP\{⟨options⟩\}{⟨entry-label⟩}{⟨inset⟩}  
modifiers: * + ⟨alt-mod⟩  

A synonym for \GLSxtrshortpl defined by the shortcuts=abbreviations package option.

\Asp\{⟨options⟩\}{⟨entry-label⟩}{⟨inset⟩}  
modifiers: * + ⟨alt-mod⟩  

A synonym for \Glsxtrshortpl defined by the shortcuts=abbreviations package option.

\asp\{⟨options⟩\}{⟨entry-label⟩}{⟨inset⟩}  
modifiers: * + ⟨alt-mod⟩  

A synonym for \glsxtrshortpl defined by the shortcuts=abbreviations package option.

B

\Beta  
glossaries-extra-bib2gls v1.27+

Defined with \providecommand, this just does \texttt{mathrm{B}}.

\bibglss@input\{⟨filename⟩\}  
glossaries-extra v1.49+

Indicates that the bib2gls records are in the file identified in the argument \texttt{(filename)}, which corresponds to the file \texttt{(basename).aux} identified in the option bibglsaux=\texttt{(basename)}.

\bibglssdualprefixlabel\{⟨label-prefix⟩\}  
\bib2gls v1.8+

Hook written to the \texttt{glstex} file identifying the dual label prefix.

\bibglssnewdualindexsymbolsecondary\{⟨label⟩\}{⟨options⟩}\{⟨name⟩\} \{⟨description⟩\}  
bib2gls

691
Command Summary

Defines secondary terms provided with \texttt{@dualindexsymbol}.

\begin{verbatim}
\bibglsprefixlabel{⟨label-prefix⟩}
\end{verbatim}
\texttt{bib2gls v1.8+} \texttt{§11.5.7; 597}

Hook written to the \texttt{glstex} file identifying the primary label prefix.

\begin{verbatim}
\bibglsetlastgrouptitle{(cs)}{(specs)}
\end{verbatim}
\texttt{bib2gls}

Sets the last group title.

\begin{verbatim}
\bibglsetlocationrecordcount{(entry-label)}{(counter)}{(location)}{(value)}
\end{verbatim}
\texttt{bib2gls v1.1+} \texttt{§11.4.2; 557}

Sets the location record count for the given entry.

\begin{verbatim}
\bibglsetrecordcount{(entry-label)}{(counter)}{(value)}
\end{verbatim}
\texttt{bib2gls v1.1+} \texttt{§11.4.2; 557}

Sets the \texttt{(counter)} record count for the given entry.

\begin{verbatim}
\bibglsettotalrecordcount{(entry-label)}{(value)}
\end{verbatim}
\texttt{bib2gls v1.1+} \texttt{§11.4.2; 556}

Sets the total record count for the given entry.

\begin{verbatim}
\bibglsttertiaryprefixlabel{⟨label-prefix⟩}
\end{verbatim}
\texttt{bib2gls v1.8+} \texttt{§11.5.7; 597}

Hook written to the \texttt{glstex} file identifying the tertiary label prefix.

\begin{verbatim}
\capitalisefmtwords{(text)}
\end{verbatim}
\texttt{mfirstuc v2.03+}

C

692
Command Summary

Converts \langle text \rangle to title case, where \langle text \rangle may contain text-block commands. The starred form only permits a text-block command at the start of the argument. Limitations apply, see the mfirstuc documentation for further details, either:

\begin{verbatim}
texdoc mfirstuc
\end{verbatim}

or visit ctan.org/pkg/mfirstuc.

\begin{verbatim}
\textbf{\texttt{\textcolor{black}{\verb|\capitalisewords{| \langle text \rangle |}}}}
mfirstuc v1.06+
\end{verbatim}

Converts \langle text \rangle to title case. Limitations apply, see the mfirstuc documentation for further details, either:

\begin{verbatim}
texdoc mfirstuc
\end{verbatim}

or visit ctan.org/pkg/mfirstuc.

\begin{verbatim}
\textbf{\texttt{\textcolor{black}{\verb|\cat\}|}}
\end{verbatim}

\textcolor{red}{\textbf{\texttt{\texttt{\textcolor{black}{\verb|§11.5.2; 561|}}}}}

\textcolor{red}{\textbf{\texttt{\textcolor{black}{\verb|\texttt{|CAT|}}}}}

Defined by \textcolor{red}{\texttt{\texttt{\textcolor{black}{\verb|\GlsXtrResourceInitEscSequences|}}}} to expand to detokenized \cat.

\begin{verbatim}
\textbf{\texttt{\textcolor{black}{\verb|\cGLS[\langle options \rangle \{\langle entry-label \rangle \}\{\langle insert \rangle \}] |}}}
\textcolor{red}{\textbf{\texttt{\texttt{\textcolor{black}{\verb|modifiers: * + \langle alt-mod \rangle|}}}}}
\end{verbatim}

\textcolor{red}{\textbf{\texttt{\texttt{\texttt{\textcolor{black}{\verb|§6.1; 314|}}}}}}

\texttt{\textcolor{black}{\verb|\cGLS[\langle options \rangle \{\langle entry-label \rangle \}\{\langle insert \rangle \}] |}}

Like \texttt{\texttt{\textcolor{black}{\verb|\GLS|}}} but hooks into the entry counting mechanism.

\begin{verbatim}
\textbf{\texttt{\textcolor{black}{\verb|\cGls[\langle options \rangle \{\langle entry-label \rangle \}\{\langle insert \rangle \}] |}}}
\textcolor{red}{\textbf{\texttt{\texttt{\textcolor{black}{\verb|modifiers: * + \langle alt-mod \rangle|}}}}}
\end{verbatim}

\textcolor{red}{\textbf{\texttt{\texttt{\texttt{\textcolor{black}{\verb|§6.1; 313|}}}}}}

\texttt{\textcolor{black}{\verb|\cGls[\langle options \rangle \{\langle entry-label \rangle \}\{\langle insert \rangle \}] |}}
glossaries v4.14+

Like \texttt{\texttt{\textcolor{black}{\verb|\Gls|}}} but hooks into the entry counting mechanism.

\begin{verbatim}
\textbf{\texttt{\textcolor{black}{\verb|\cglsl[\langle options \rangle \{\langle entry-label \rangle \}\{\langle insert \rangle \}] |}}}
\textcolor{red}{\textbf{\texttt{\texttt{\textcolor{black}{\verb|modifiers: * + \langle alt-mod \rangle|}}}}}
\end{verbatim}

\textcolor{red}{\textbf{\texttt{\texttt{\texttt{\textcolor{black}{\verb|§6.1; 312|}}}}}}

\texttt{\textcolor{black}{\verb|\cglsl[\langle options \rangle \{\langle entry-label \rangle \}\{\langle insert \rangle \}] |}}
glossaries v4.14+

Like \texttt{\texttt{\textcolor{black}{\verb|\gls|}}} but hooks into the entry counting mechanism.

693
Format used by `\cGLS` if the entry was not used more than the given trigger value on the previous run.

Format used by `\cGls` if the entry was not used more than the given trigger value on the previous run.

Format used by `\cgls` if the entry was not used more than the given trigger value on the previous run.

Like `\GLSp1` but hooks into the entry counting mechanism.

Like `\Glspl` but hooks into the entry counting mechanism.

Like `\glspl` but hooks into the entry counting mechanism.
Format used by \cGLSpl if the entry was not used more than the given trigger value on the previous run.

\cGLSplformat{⟨entry-label⟩}{⟨insert⟩} \hspace{1em} glossaries v4.14+

§6.1; 314

Format used by \cglspl if the entry was not used more than the given trigger value on the previous run.

\cglsplformat{⟨entry-label⟩}{⟨insert⟩} \hspace{1em} glossaries v4.14+

§6.1; 313

Format used by \cglspl if the entry was not used more than the given trigger value on the previous run.

\Chi \hspace{1em} glossaries-extra-bib2gls v1.27+

§11.5.8; 599

Defined with \providecommand, this just does \mathrm{X}.

\CS

§11.5.2; 561

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \CS.

\cs{⟨csname⟩}

§11.5.2; 561

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \csname.

\csGlsXtrLetField{⟨entry-label⟩}{⟨field-label⟩}{⟨cs-name⟩} \hspace{1em} glossaries-extra v1.12+

§3.5; 40

Like \GlsXtrLetField but internally uses (etoolbox’s) \csletcs instead of \cslet.

\CurrentGlossary \hspace{1em} glossaries v3.0+

§3.5; 40
Defined by the \print...glossary commands to the current glossary label.

\CustomAbbreviationFields

Expands to the default field definitions for the entry.

D
\defglsentryfmt[\langle type\rangle]{\langle display\rangle}
glossaries

Overrides the default display format (\glsentryfmt) for the given glossary. If \langle type\rangle is omitted, \glsdefaulttype is assumed. This will make the \gls-like commands do \langle display\rangle for any entries that have the type field set to the given \langle type\rangle. If you want to support any abbreviation styles, you need to include \glssetabbrvfmt in \langle display\rangle. Non-regular abbreviation styles are designed to work with \glsxtrgenabbrvfmt.

\delimN
glossaries

Used as a separator between locations.

\delimR
glossaries

Used between the start and end of a location range.

\descriptionname
\(\text{initial: Description}\)
(language-sensitive)
glossaries

Expands to the title of the description column for headed tabular-like styles.

\dGLS[\langle options\rangle]{\langle entry-label\rangle}[\langle insert\rangle]
\textit{modifiers: * + \langle alt-mod\rangle}
glossaries-extra-bib2gls v1.37+

As \dgls but uses \GLS.
Command Summary

\texttt{\textbackslash dGls}\langle\textit{options}\rangle\{\langle\text{entry-label}\rangle}\{\langle\text{insert}\rangle\}

\begin{itemize}
  \item modifiers: * + \langle\text{alt-mod}\rangle
\end{itemize}

\texttt{\textbackslash dGls}\langle\textit{options}\rangle\{\langle\text{entry-label}\rangle\}{\langle\text{insert}\rangle}

\begin{itemize}
  \item modifiers: * + \langle\text{alt-mod}\rangle
\end{itemize}

As \texttt{\textbackslash dglS} but uses \texttt{\Gls}.

\texttt{\textbackslash dGls}\langle\textit{options}\rangle\{\langle\text{prefix}\rangle\langle\text{entry-label}\rangle\}\{\langle\text{insert}\rangle\}

\begin{itemize}
  \item modifiers: * + \langle\text{alt-mod}\rangle
\end{itemize}

Does \texttt{\textbackslash glS}\langle\textit{options}\rangle\{\langle\text{prefix}\rangle\langle\text{entry-label}\rangle\}\{\langle\text{insert}\rangle\} for the first prefix in the prefix list that matches a defined entry.

\texttt{\textbackslash dGLSdisp}\langle\textit{options}\rangle\{\langle\text{entry-label}\rangle\}\{\langle\text{text}\rangle\}

\begin{itemize}
  \item modifiers: * + \langle\text{alt-mod}\rangle
\end{itemize}

As \texttt{\textbackslash dglSdisp} but applies sentence case.

\texttt{\textbackslash dGLSdisp}\langle\textit{options}\rangle\{\langle\text{entry-label}\rangle\}\{\langle\text{text}\rangle\}

\begin{itemize}
  \item modifiers: * + \langle\text{alt-mod}\rangle
\end{itemize}

As \texttt{\textbackslash dglSdisp} but uses \texttt{\glSdisp}.

\texttt{\textbackslash dGLSfield}\langle\textit{options}\rangle\{\langle\text{entry-label}\rangle\}\{\langle\text{field-label}\rangle\}\{\langle\text{text}\rangle\}

\begin{itemize}
  \item modifiers: * + \langle\text{alt-mod}\rangle
\end{itemize}

As \texttt{\textbackslash dglSfield} but all caps.

\texttt{\textbackslash dGLSfield}\langle\textit{options}\rangle\{\langle\text{entry-label}\rangle\}\{\langle\text{field-label}\rangle\}\{\langle\text{text}\rangle\}

\begin{itemize}
  \item modifiers: * + \langle\text{alt-mod}\rangle
\end{itemize}

As \texttt{\textbackslash dglSfield} but applies sentence case.
As \dglsl but selects the first matching label that has an entry with the field set.

\texttt{\dglslactualfieldlabel} \hspace{1cm} \textit{glossaries-extra-bib2gls v1.49+} \hspace{1cm} \S11.5.7; 593

Set by the \dglsl family of commands to the actual field used. This will either be the requested field or the fallback field.

\texttt{\dglslcurrentfieldlabel} \hspace{1cm} \textit{glossaries-extra-bib2gls v1.49+} \hspace{1cm} \S11.5.7; 593

Set by the \dglsl family of commands to the given \textit{⟨field-label⟩}.

\texttt{\dglslfallbackfieldlabel} \hspace{1cm} \textit{initial: text} \hspace{1cm} \textit{glossaries-extra-bib2gls v1.49+} \hspace{1cm} \S11.5.7; 593

Expands to the fallback field to use for the \dglsl family of commands.

\texttt{\dGlsl}[\textit{⟨options⟩}]{\textit{⟨entry-label⟩}}{\textit{⟨text⟩}} \hspace{1cm} \textit{modifiers: * + ⟨alt-mod⟩} \hspace{1cm} \S11.5.7; 591

As \dglsl but applies sentence case.

\texttt{\dGlsl}[\textit{⟨options⟩}]{\textit{⟨entry-label⟩}}{\textit{⟨text⟩}} \hspace{1cm} \textit{modifiers: * + ⟨alt-mod⟩} \hspace{1cm} \S11.5.7; 591

As \dglsl but uses \glsl.

\texttt{\GLSpl}[\textit{⟨options⟩}]{\textit{⟨entry-label⟩}}{\textit{⟨insert⟩}} \hspace{1cm} \textit{modifiers: * + ⟨alt-mod⟩} \hspace{1cm} \S11.5.7; 591

As \dglsl but uses \GLSpl.

\texttt{\GLSpl}[\textit{⟨options⟩}]{\textit{⟨entry-label⟩}}{\textit{⟨insert⟩}} \hspace{1cm} \textit{modifiers: * + ⟨alt-mod⟩} \hspace{1cm} \S11.5.7; 591
As \texttt{\textbackslash dgls} but uses \texttt{\textbackslash Glsp}.  

\begin{verbatim}
\dglspl\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle insert\rangle\}  
modifers: * + \langle alt-mod\rangle
\end{verbatim}

§11.5.7; 590

As \texttt{\textbackslash dgls} but uses \texttt{\textbackslash Glsp}.  

\begin{verbatim}
\Digamma
\end{verbatim}

§11.5.8; 599

Defined with \texttt{\textbackslash providecommand}, this just does \texttt{\textbackslash mathrm\{F\}}.  

\begin{verbatim}
\doiglossarynoexistsordo\{\langle glossary-type\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}
\end{verbatim}

§11.5.8; 599

If the glossary given by \texttt{\langle glossary-type\rangle} doesn’t exist, this does \texttt{\langle true\rangle}, otherwise it generates an error and does \texttt{\langle false\rangle}. This uses the starred form of \texttt{\textbackslash ifglossaryexists} to test for existence.

\begin{verbatim}
\DTLformatlist\{\langle csv-list\rangle\}
\end{verbatim}

§11.5.8; 599

Formats the comma-separated list \texttt{\langle csv-list\rangle}. One-level expansion is performed on \texttt{\langle csv-list\rangle}. See the \texttt{datatool} documentation for further details, either:  

\begin{verbatim}
texdoc datatool
\end{verbatim}

or visit ctan.org/pkg/datatool.

\begin{verbatim}
\DTLifinlist\{\langle element\rangle\}\{\langle csv-list\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}
\end{verbatim}

§11.5.8; 599

Does \texttt{\langle true\rangle} if \texttt{\langle element\rangle} is contained in the comma-separated list \texttt{\langle csv-list\rangle}, otherwise does \texttt{\langle false\rangle}. One-level expansion is performed on \texttt{\langle csv-list\rangle}, but not on \texttt{\langle element\rangle}. See the \texttt{datatool} documentation for further details, either:  

\begin{verbatim}
texdoc datatool
\end{verbatim}
Command Summary

or visit ctan.org/pkg/datatool.

E

\texttt{\textsc{eglsetwidest}}[⟨level⟩]{⟨name⟩} \hspace{1cm} glossaries-extra-stylemods v1.05+

§8.6.5.4; 439

As \texttt{\textsc{glssetwidest}} but expands ⟨text⟩.

\texttt{\textsc{eglupdatewidest}}[⟨level⟩]{⟨name⟩} \hspace{1cm} glossaries-extra-stylemods v1.23+

§8.6.5.4; 440

As \texttt{\textsc{glsupdatewidest}} but expands ⟨name⟩.

\texttt{\textsc{eGlsXtrSetField}}{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩} \hspace{1cm} glossaries-extra v1.12+

§3.5; 40

As \texttt{\textsc{GlsXtrSetField}} but expands the value.

\texttt{\textsc{entryname}} \hspace{1cm} initial: Notation glossaries

Expands to the title of the name column for headed tabular-like styles.

\texttt{\textsc{Epsilon}} \hspace{1cm} glossaries-extra-bib2gls v1.27+

§11.5.8; 599

Defined with \texttt{\textsc{providecommand}}, this just does \texttt{\textsc{mathrm}{E}}.

\texttt{\textsc{Eta}} \hspace{1cm} glossaries-extra-bib2gls v1.27+

§11.5.8; 599

Defined with \texttt{\textsc{providecommand}}, this just does \texttt{\textsc{mathrm}{H}}.

\texttt{\textsc{ExtraCustomAbbreviationFields}} \hspace{1cm} glossaries-extra v1.31+

§4.5.3.1; 164

Expands to additional fields that need to be set with \texttt{\textsc{newabbreviation}}.
Package: glossaries v1.14+

\firstacronymfont{⟨text⟩}

Used to encapsulate the acronym short form on first use by the base glossaries package. This is redefined by glossaries-extra to use \glsfirstabbrvfont.

\FIRSTLC

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \FIRSTLC.

\FIRSTUC

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \FIRSTUC.

\forallabbreviationlists{⟨cs⟩}{⟨body⟩}

glossaries-extra v1.42+

Iterates overall all lists of abbreviations, defines the command ⟨cs⟩ to the current label and does ⟨body⟩.

\forallacronyms{⟨cs⟩}{⟨body⟩}

glossaries v4.08+

Iterates overall all glossaries that have been declared lists of acronyms, defines the command ⟨cs⟩ to the current label and does ⟨body⟩. Use \forallabbreviationlists with glossaries-extra.

\forallglossaries[⟨types⟩]{⟨cs⟩}{⟨body⟩}

glossaries

Iterates overall all the glossary labels given in the ⟨types⟩ argument, defines the command ⟨cs⟩ to the current label and does ⟨body⟩. If the optional argument is omitted, the list of all non-ignored glossaries is assumed.

\forallglsentries[⟨types⟩]{⟨cs⟩}{⟨body⟩}

glossaries
Does \texttt{\forglsentries} for each glossary. The optional argument \texttt{\langle types \rangle} is a comma-separated list of glossary labels. If omitted, all non-ignored glossaries is assumed.

\texttt{\forglsentries\{\langle type \rangle\}\{\langle cs \rangle\}\{\langle body \rangle\}}

Iterates over all entries in the given glossary and, at each iteration, defines the command \texttt{\langle cs \rangle} to the current entry label and does \texttt{\langle body \rangle}. The optional argument \texttt{\langle type \rangle} is the glossary label and defaults to \texttt{\glsdefaulttype} if omitted. This command can’t be used with \texttt{\bib2gls} since there are no defined entries until \texttt{\bib2gls} has selected them and added them to the \texttt{glstex} file.

\textbf{G}

\texttt{\glssetwidest\{\langle level \rangle\}\{\langle name \rangle\}} \hspace{2cm} \text{glossaries-extra-stylemods v1.21+}

As \texttt{\glssetwidest} but global.

\texttt{\glsupdatewidest\{\langle level \rangle\}\{\langle name \rangle\}} \hspace{2cm} \text{glossaries-extra-stylemods v1.23+}

As \texttt{\glsupdatewidest} but global.

\texttt{\GlsXtrSetField\{\langle entry-label \rangle\}\{\langle field-label \rangle\}\{\langle value \rangle\}} \hspace{2cm} \text{glossaries-extra v1.12+}

As \texttt{\GlsXtrSetField} but globally assigns the value.

\textbf{Glo}

\texttt{\glolinkprefix} \hspace{2cm} \textit{initial: glo:} \hspace{2cm} \text{glossaries}

Expands to the default prefix for the entry's hypertarget anchor in the glossary.

\texttt{\GlossariesAbbrStyleTooComplexWarning} \hspace{2cm} \text{glossaries-extra v1.49+}

702
Issues a warning with \GlossariesExtraWarning when a command is used that isn’t supported by a complex abbreviation style.

\GlossariesExtraInfo{⟨message⟩}  
glossaries-extra v1.51+

Writes an information message to the transcript.

\glossariesextrasetup{⟨options⟩}  
§2; 8

Change allowed options that are defined or modified by the glossaries-extra package. Note that some options can only be passed as package options.

\GlossariesExtraWarning{⟨message⟩}  

Writes a warning in the transcript with the current line number. The nowarn option redefines this command to do nothing.

\GlossariesExtraWarningNoLine{⟨message⟩}  
glossaries-extra

Writes a warning in the transcript without a corresponding line number. The nowarn option redefines this command to do nothing.

\glossaryentrynumbers{⟨location list⟩}  
glossaries

Used within the glossary to encapsulate the location list (redefined by the nonumberlist option).

\glossaryheader  
glossaries

Inserted after \begin{theglossary}. This command should be redefined by the glossary style.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glossaryname</td>
<td>initial: Glossary (language-sensitive)</td>
</tr>
<tr>
<td>\glossarypostamble</td>
<td>Used at the end of the glossary.</td>
</tr>
<tr>
<td>\glossarypreamble</td>
<td>Used at the start of the glossary. This will be locally redefined to</td>
</tr>
<tr>
<td></td>
<td>the preamble associated with the current glossary, if one has been</td>
</tr>
<tr>
<td></td>
<td>set.</td>
</tr>
<tr>
<td>\glossarysection</td>
<td>Occurs at the start of a glossary (except with \printunrtinnerglossary). This will typically be defined to use a sectioning command, such as \section or \chapter. The default definition follows the section and numberedsection options.</td>
</tr>
<tr>
<td>\glossarytitle</td>
<td>initialised by the \print...glossary set of commands (such as \printglossary) to the current glossary’s title.</td>
</tr>
<tr>
<td>\glossarytoctitle</td>
<td>initialised by the \print...glossary set of commands (such as \printglossary) to the current glossary’s table of contents title.</td>
</tr>
<tr>
<td>\glossentry</td>
<td>used to format a top-level entry. This command should be redefined by the glossary style.</td>
</tr>
</tbody>
</table>

Expands to the default glossary title (provided by glossaries if not already defined).
<table>
<thead>
<tr>
<th>Command Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\Glossentrydesc{⟨entry-label⟩}</td>
<td>\glossaries v3.08a+</td>
</tr>
<tr>
<td>As \glossentrydesc but sentence case.</td>
<td></td>
</tr>
<tr>
<td>\glossentrydesc{⟨entry-label⟩}</td>
<td>\glossaries v3.08a+</td>
</tr>
<tr>
<td>Used by glossary styles to display the entry’s description.</td>
<td></td>
</tr>
<tr>
<td>\Glossentryname{⟨entry-label⟩}</td>
<td>\glossaries v3.08a+</td>
</tr>
<tr>
<td>As \glossentryname but sentence case.</td>
<td></td>
</tr>
<tr>
<td>\glossentryname{⟨entry-label⟩}</td>
<td>\glossaries v3.08a+</td>
</tr>
<tr>
<td>Used by glossary styles to display the entry’s name.</td>
<td></td>
</tr>
<tr>
<td>\glossentrynameother{⟨entry-label⟩}{⟨field-label⟩}</td>
<td>\glossaries-extra v1.22+</td>
</tr>
<tr>
<td>Behaves like \glossentryname but uses the given field (identified by its internal label) instead of name.</td>
<td>§8.6; 425</td>
</tr>
<tr>
<td>\Glossentrysymbol{⟨entry-label⟩}</td>
<td>\glossaries v3.08a+</td>
</tr>
<tr>
<td>As \glossentrysymbol but sentence case.</td>
<td></td>
</tr>
<tr>
<td>\glossentrysymbol{⟨entry-label⟩}</td>
<td>\glossaries v3.08a+</td>
</tr>
<tr>
<td>Used by glossary styles to display the entry’s symbol.</td>
<td></td>
</tr>
<tr>
<td>\glossxtrsetpopts</td>
<td>\glossaries-extra v1.07+</td>
</tr>
<tr>
<td>§5.4; 231</td>
<td></td>
</tr>
<tr>
<td>705</td>
<td></td>
</tr>
</tbody>
</table>
Command Summary

Used at the start of each glossary to set the current options for the \glsxtrp set of commands (with \glsxtrsetpopts).

**\GLS**

\GLS[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]  
**modifiers:** * + ⟨alt-mod⟩  
glossaries

As \gls but converts the link text to all caps.

\Gls[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]  
**modifiers:** * + ⟨alt-mod⟩  
glossaries

As \gls but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc.

\gls[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩]  
**modifiers:** * + ⟨alt-mod⟩  
glossaries

References the entry identified by ⟨entry-label⟩. The text produced depends on whether or not this is the first use and whether or not the regular attribute has been set. The ⟨insert⟩ argument may be inserted at the end of the link text or may be inserted at a different point (for example, after the long form on first use for some abbreviation styles). For the first optional argument, see \glslink options.

\glsabbrvdefaultfont{⟨text⟩}

Formatting command for the short form used by the abbreviation styles that don’t apply a font change by default.

\glsabbrvemfont{⟨text⟩}  
glossaries-extra v1.04+

Short form font used by the “em” abbreviation styles.

\glsabbrvfont{⟨text⟩}
Command Summary

Font formatting command for the short form, initialised by the abbreviation style.

\glsabbrvhyphenfont{⟨text⟩}  \hspace{1cm} \texttt{glossaries-extra v1.17+}  \hspace{1cm} 148

Short form font used by the “hyphen” abbreviation styles.

\glsabbrvonlyfont{⟨text⟩}  \hspace{1cm} \texttt{glossaries-extra v1.17+}  \hspace{1cm} 154

Short form font used by the “only” abbreviation styles.

\glsabbrvscfont{⟨text⟩}  \hspace{1cm} \texttt{glossaries-extra v1.17+}  \hspace{1cm} 156

Short form font used by the small caps “sc” abbreviation styles.

\glsabbrvsconlyfont{⟨text⟩}  \hspace{1cm} \texttt{glossaries-extra v1.48+}  \hspace{1cm} 154

Short form font used by the “sc-only” styles, such as \texttt{long-only-short-sc-only}.

\glsabbrvscuserfont{⟨text⟩}  \hspace{1cm} \texttt{glossaries-extra v1.48+}  \hspace{1cm} 136

Short form font used by the small caps “sc-user” abbreviation styles.

\glsabbrvsmfont{⟨text⟩}  \hspace{1cm} \texttt{glossaries-extra v1.17+}  \hspace{1cm} 157

Short form font used by the “sm” abbreviation styles.

\glsabbrvuserfont{⟨text⟩}  \hspace{1cm} \texttt{glossaries-extra v1.04+}  \hspace{1cm} 135

Short form font used by the “user” abbreviation styles.
Command Summary

\texttt{\glsabspace}\{\textit{label}\}\} \hspace{1cm} \texttt{glossaries-extra v1.49+}

As \texttt{\glsacspace} but includes inner formatting. Unlike \texttt{\glsacspace}, this command is robust.

\texttt{\GLSaccessdesc}\{\textit{entry-label}\}\} \hspace{1cm} §9.2; 505

The all caps version of \texttt{\glsaccessdesc}.

\texttt{\Glsaccessdesc}\{\textit{entry-label}\}\} \hspace{1cm} §9.2; 505

The sentence case version of \texttt{\glsaccessdesc}.

\texttt{\glsaccessdesc}\{\textit{entry-label}\}\} \hspace{1cm} §9.2; 505

If accessibility support was enabled when \texttt{glossaries-extra} was loaded (\texttt{accsupp}) this will display the value of the \texttt{description} key with the accessibility support enabled for that key (\texttt{descriptionaccess}). If there is no accessibility support, this just uses \texttt{\glsentrydesc}.

\texttt{\GLSaccessdescplural}\{\textit{entry-label}\}\} \hspace{1cm} §9.2; 505

The all caps version of \texttt{\glsaccessdescplural}.

\texttt{\Glsaccessdescplural}\{\textit{entry-label}\}\} \hspace{1cm} §9.2; 505

The sentence case version of \texttt{\glsaccessdescplural}.

\texttt{\glsaccessdescplural}\{\textit{entry-label}\}\} \hspace{1cm} §9.2; 505

If accessibility support was enabled when \texttt{glossaries-extra} was loaded (\texttt{accsupp}) this will display the value of the \texttt{descriptionplural} key with the accessibility support enabled for
that key (descriptionpluralaccess). If there is no accessibility support, this just uses \glsentrydescplural.

\GLSaccessfirst{\langle entry-label \rangle}

§9.2; 503

The all caps version of \glsaccessfirst.

\Glsaccessfirst{\langle entry-label \rangle}

§9.2; 503

The sentence case version of \glsaccessfirst.

\glsaccessfirst{\langle entry-label \rangle}

§9.2; 503

If accessibility support was enabled when glossaries-extra was loaded (acccessupp) this will display the value of the first key with the accessibility support enabled for that key (firstaccess). If there is no accessibility support, this just uses \glsentryfirst.

\GLSaccessfirstplural{\langle entry-label \rangle}

§9.2; 504

The all caps version of \glsaccessfirstplural.

\Glsaccessfirstplural{\langle entry-label \rangle}

§9.2; 504

The sentence case version of \glsaccessfirstplural.

\glsaccessfirstplural{\langle entry-label \rangle}

§9.2; 504

If accessibility support was enabled when glossaries-extra was loaded (acccessupp) this will display the value of the firstplural key with the accessibility support enabled for that key (firstpluralaccess). If there is no accessibility support, this just uses \glsentryfirstplural.
Similar to \texttt{\GLSaccessdesc} but formats the displayed text with \texttt{\GLSfmtfield}.

\begin{mdframed}
\texttt{\GLSaccessfmtdesc}\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\} \quad \text{glossaries-extra v1.49+}
\end{mdframed}

\textit{§9.3; 512}

Similar to \texttt{\Glsaccessdesc} but formats the displayed text with \texttt{\Glsfmtfield}.

\begin{mdframed}
\texttt{\Glsaccessfmtdesc}\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\} \quad \text{glossaries-extra v1.49+}
\end{mdframed}

\textit{§9.3; 512}

Similar to \texttt{\glsaccessdesc} but formats the displayed text with \texttt{\glsfmtfield}.

\begin{mdframed}
\texttt{\glsaccessfmtdesc}\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\} \quad \text{glossaries-extra v1.49+}
\end{mdframed}

\textit{§9.3; 512}

Similar to \texttt{\GLSaccessfmtdescplural} but formats the displayed text with \texttt{\GLSfmtfield}.

\begin{mdframed}
\texttt{\GLSaccessfmtdescplural}\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\} \quad \text{glossaries-extra v1.49+}
\end{mdframed}

\textit{§9.3; 513}

Similar to \texttt{\Glsaccessfmtdescplural} but formats the displayed text with \texttt{\Glsfmtfield}.

\begin{mdframed}
\texttt{\Glsaccessfmtdescplural}\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\} \quad \text{glossaries-extra v1.49+}
\end{mdframed}

\textit{§9.3; 513}

Similar to \texttt{\glsaccessfmtdescplural} but formats the displayed text with \texttt{\glsfmtfield}.

\begin{mdframed}
\texttt{\glsaccessfmtdescplural}\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\} \quad \text{glossaries-extra v1.49+}
\end{mdframed}

\textit{§9.3; 512}

Similar to \texttt{\GLSaccessfmtfirst} but formats the displayed text with \texttt{\GLSfmtfield}.

\begin{mdframed}
\texttt{\GLSaccessfmtfirst}\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\} \quad \text{glossaries-extra v1.49+}
\end{mdframed}

\textit{§9.3; 511}

Similar to \texttt{\Glsaccessfmtfirst} but formats the displayed text with \texttt{\Glsfmtfield}.

\begin{mdframed}
\texttt{\Glsaccessfmtfirst}\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\} \quad \text{glossaries-extra v1.49+}
\end{mdframed}

\textit{§9.3; 511
Command Summary

Similar to \texttt{\Glsaccessfirst} but formats the displayed text with \texttt{\Glsfmtfield}.

\begin{verbatim}
\Glsaccessfmtfirst{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
\end{verbatim}

§9.3; 511

Similar to \texttt{\glsaccessfmtfirst} but formats the displayed text with \texttt{\glsfmtfield}.

\begin{verbatim}
\GLSaccessfmtfirstplural{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
\end{verbatim}

§9.3; 511

Similar to \texttt{\GLSaccessfmtfirstplural} but formats the displayed text with \texttt{\GLSfmtfield}.

\begin{verbatim}
\Glsaccessfmtfirstplural{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
\end{verbatim}

§9.3; 511

Similar to \texttt{\Glsaccessfmtfirstplural} but formats the displayed text with \texttt{\Glsfmtfield}.

\begin{verbatim}
\glsaccessfmtfirstplural{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
\end{verbatim}

§9.3; 511

Similar to \texttt{\Glsaccessfmtlong} but formats the displayed text with \texttt{\Glsfmtfield}.

\begin{verbatim}
\GLSaccessfmtlong{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
\end{verbatim}

§9.3; 514

Similar to \texttt{\GLSaccessfmtlong} but formats the displayed text with \texttt{\GLSfmtfield}.

\begin{verbatim}
\Glsaccessfmtlong{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
\end{verbatim}

§9.3; 514

Similar to \texttt{\Glsaccessfmtlong} but formats the displayed text with \texttt{\Glsfmtfield}.

\begin{verbatim}
\glsaccessfmtlong{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
\end{verbatim}

§9.3; 514

Similar to \texttt{\glsaccessfmtlong} but formats the displayed text with \texttt{\glsfmtfield}.
Command Summary

\texttt{\textbackslash GLSaccessfmtlongpl\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}}

Similar to \texttt{\textbackslash GLSaccesslongpl} but formats the displayed text with \texttt{\textbackslash GLSfmtfield}.

\texttt{\textbackslash Glsaccessfmtlongpl\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}}

Similar to \texttt{\textbackslash Glsaccesslongpl} but formats the displayed text with \texttt{\textbackslash Glsfmtfield}.

\texttt{\textbackslash glsaccessfmtlongpl\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}}

Similar to \texttt{\textbackslash glsaccesslongpl} but formats the displayed text with \texttt{\textbackslash glsfmtfield}.

\texttt{\textbackslash GLSaccessfmtname\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}}

Similar to \texttt{\textbackslash GLSaccessname} but formats the displayed text with \texttt{\textbackslash GLSfmtfield}.

\texttt{\textbackslash Glsaccessfmtname\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}}

Similar to \texttt{\textbackslash Glsaccessname} but formats the displayed text with \texttt{\textbackslash Glsfmtfield}.

\texttt{\textbackslash glsaccessfmtname\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}}

Similar to \texttt{\textbackslash glsaccessname} but formats the displayed text with \texttt{\textbackslash glsfmtfield}.

\texttt{\textbackslash GLSaccessfmtplural\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}}

Similar to \texttt{\textbackslash GLSaccessplural} but formats the displayed text with \texttt{\textbackslash GLSfmtfield}.

\texttt{\textbackslash Glsaccessfmtplural\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}}

Similar to \texttt{\textbackslash Glsaccessplural} but formats the displayed text with \texttt{\textbackslash Glsfmtfield}.

\texttt{\textbackslash glsaccessfmtplural\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}}

Similar to \texttt{\textbackslash glsaccessplural} but formats the displayed text with \texttt{\textbackslash glsfmtfield}.
Command Summary

Similar to `\Glsaccessplural` but formats the displayed text with `\Glsfmtfield`.

\glsaccessfmtplural{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}  
glossaries-extra v1.49+  
§9.3; 510

Similar to `\glsaccessplural` but formats the displayed text with `\glsfmtfield`.

\GLSaccessfmtshort{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}  
glossaries-extra v1.49+  
§9.3; 513

Similar to `\GLSaccessshort` but formats the displayed text with `\GLSfmtfield`.

\GLSaccessfmtshort{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}  
glossaries-extra v1.49+  
§9.3; 513

Similar to `\GLSaccessshort` but formats the displayed text with `\GLSfmtfield`.

\glsaccessfmtshort{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}  
glossaries-extra v1.49+  
§9.3; 513

Similar to `\glsaccessshort` but formats the displayed text with `\glsfmtfield`.

\GLSaccessfmtshortpl{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}  
glossaries-extra v1.49+  
§9.3; 513

Similar to `\GLSaccessshortpl` but formats the displayed text with `\GLSfmtfield`.

\GLSaccessfmtshortpl{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}  
glossaries-extra v1.49+  
§9.3; 513

Similar to `\GLSaccessshortpl` but formats the displayed text with `\GLSfmtfield`.

\glsaccessfmtshortpl{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}  
glossaries-extra v1.49+  
§9.3; 513

Similar to `\glsaccessshortpl` but formats the displayed text with `\glsfmtfield`.

713
Similar to \GLSaccesssym but formats the displayed text with \GLSfmtfield.

Similar to \Glsaccesssym but formats the displayed text with \Glsfmtfield.

Similar to \glsaccesssym but formats the displayed text with \glsfmtfield.

Similar to \GLSaccesssymplural but formats the displayed text with \GLSfmtfield.

Similar to \Glsaccesssymplural but formats the displayed text with \Glsfmtfield.

Similar to \glsaccesssymplural but formats the displayed text with \glsfmtfield.

Similar to \GLSaccessfmttext but formats the displayed text with \GLSfmtfield.

Similar to \Glsaccessfmttext but formats the displayed text with \Glsfmtfield.
Command Summary

Similar to \Glsaccessstext but formats the displayed text with \Glsfmtfield.

\glsaccessfmttext\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 510

Similar to \glsaccessstext but formats the displayed text with \glsfmtfield.

\GLSaccessfmtuseri\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 515

Similar to \GLSaccessuseri but formats the displayed text with \GLSfmtfield.

\GLSaccessfmtuseri\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 514

Similar to \GLSaccessuseri but formats the displayed text with \GLSfmtfield.

\GLSaccessfmtuseri\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 514

Similar to \GLSaccessuseri but formats the displayed text with \GLSfmtfield.

\GLSaccessfmtuseri\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 515

Similar to \GLSaccessuserii but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtuserii\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 515

Similar to \Glsaccessuserii but formats the displayed text with \Glsfmtfield.

\Glsaccessfmtuserii\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 515

Similar to \Glsaccessuserii but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuserii\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 515

Similar to \glsaccessuserii but formats the displayed text with \glsfmtfield.

\glsaccessfmtuserii\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 515

Similar to \glsaccessuserii but formats the displayed text with \glsfmtfield.

\glsaccessfmtuserii\{⟨insert⟩\}{⟨cs⟩}{⟨entry-label⟩} glossaries-extra v1.49+

§9.3; 515

715
Command Summary

\GLSaccessfmtuseri\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}  glossaries-extra v1.49+

§9.3; 515

Similar to \GLSaccessuseri but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtuseri\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}  glossaries-extra v1.49+

§9.3; 515

Similar to \Glsaccessuseri but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuseri\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}  glossaries-extra v1.49+

§9.3; 515

Similar to \glsaccessuseri but formats the displayed text with \glsfmtfield.

\GLSaccessfmtuseriv\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}  glossaries-extra v1.49+

§9.3; 516

Similar to \GLSaccessuseriv but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtuseriv\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}  glossaries-extra v1.49+

§9.3; 516

Similar to \Glsaccessuseriv but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuseriv\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}  glossaries-extra v1.49+

§9.3; 515

Similar to \glsaccessuseriv but formats the displayed text with \glsfmtfield.

\GLSaccessfmtusers\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}  glossaries-extra v1.49+

§9.3; 516

Similar to \GLSaccessusers but formats the displayed text with \GLSfmtfield.

\Glsaccessfmtusers\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}  glossaries-extra v1.49+

§9.3; 516

Similar to \Glsaccessusers but formats the displayed text with \Glsfmtfield.

\glsaccessfmtusers\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}  glossaries-extra v1.49+

§9.3; 515

Similar to \glsaccessusers but formats the displayed text with \glsfmtfield.
Command Summary

Similar to \Glsaccessuserv but formats the displayed text with \Glsfmtfield.

\glsaccessfmtuserv{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
§9.3; 516

Similar to \glsaccessuserv but formats the displayed text with \Glsfmtfield.

\GLSaccessfmtuservi{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
§9.3; 516

Similar to \GLSaccessuservi but formats the displayed text with \GLSfmtfield.

\GLSaccessfmtuservi{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
§9.3; 516

Similar to \GLSaccessuservi but formats the displayed text with \GLSfmtfield.

\glsaccessfmtuservi{⟨insert⟩}{⟨cs⟩}{⟨entry-label⟩}
glossaries-extra v1.49+
§9.3; 516

Similar to \glsaccessuservi but formats the displayed text with \glsfmtfield.

\glsaccessibility{⟨options⟩}{⟨PDF element⟩}{⟨value⟩}{⟨content⟩}
glossaries-accsupp v4.45+

Applies ⟨value⟩ as the accessibility attribute ⟨PDF element⟩ for the given ⟨content⟩. This internally uses the accessibility support provided by accsupp.

\GLSaccesslong{⟨entry-label⟩}
§9.2; 506

The all caps version of \glsaccesslong.

\Glsaccesslong{⟨entry-label⟩}
§9.2; 506

The sentence case version of \glsaccesslong.
\texttt{\textbackslash glsaccesslong\{\langle entry-label\rangle\}}

If accessibility support was enabled when glossaries-extra was loaded (\texttt{accsupp}) this will display the value of the \texttt{long} key with the accessibility support enabled for that key (\texttt{longaccess}). If there is no accessibility support, this just uses \texttt{\textbackslash glsentrylong}.

\texttt{\textbackslash GLSaccesslongpl\{\langle entry-label\rangle\}}

The all caps version of \texttt{\textbackslash glsaccesslongpl}.

\texttt{\textbackslash Glsaccesslongpl\{\langle entry-label\rangle\}}

The sentence case version of \texttt{\textbackslash glsaccesslongpl}.

\texttt{\textbackslash glsaccesslongpl\{\langle entry-label\rangle\}}

If accessibility support was enabled when glossaries-extra was loaded (\texttt{accsupp}) this will display the value of the \texttt{longplural} key with the accessibility support enabled for that key (\texttt{longpluralaccess}). If there is no accessibility support, this just uses \texttt{\textbackslash glsentrylongpl}.

\texttt{\textbackslash GLSaccessname\{\langle entry-label\rangle\}}

The all caps version of \texttt{\textbackslash glsaccessname}.

\texttt{\textbackslash Glsaccessname\{\langle entry-label\rangle\}}

The sentence case version of \texttt{\textbackslash glsaccessname}.

\texttt{\textbackslash glsaccessname\{\langle entry-label\rangle\}}

If accessibility support was enabled when glossaries-extra was loaded (\texttt{accsupp}) this will display the value of the \texttt{name} key with the accessibility support enabled for that key (\texttt{access}). If there is no accessibility support, this just uses \texttt{\textbackslash glsentryname}.
\GLSaccessplural{⟨entry-label⟩}

The all caps version of \glsaccessplural.

\Glsaccessplural{⟨entry-label⟩}

The sentence case version of \glsaccessplural.

\glsaccessplural{⟨entry-label⟩}

If accessibility support was enabled when glossaries-extra was loaded (accsupp) this will display the value of the plural key with the accessibility support enabled for that key (plural-access). If there is no accessibility support, this just uses \glsentryplural.

\GLSaccessshort{⟨entry-label⟩}

The all caps version of \glsaccessshort.

\Glsaccessshort{⟨entry-label⟩}

The sentence case version of \glsaccessshort.

\glsaccessshort{⟨entry-label⟩}

If accessibility support was enabled when glossaries-extra was loaded (accsupp) this will display the value of the short key with the accessibility support enabled for that key (short-access). If there is no accessibility support, this just uses \glsentryshort.

\GLSaccessshortpl{⟨entry-label⟩}

The all caps version of \glsaccessshortpl.
The sentence case version of `\glsaccessshortpl`.

If accessibility support was enabled when `glossaries-extra` was loaded (`accsupp`) this will display the value of the `shortplural` key with the accessibility support enabled for that key (`shortpluralaccess`). If there is no accessibility support, this just uses `\glsentryshortpl`.

The all caps version of `\glsaccesssymbol`.

If accessibility support was enabled when `glossaries-extra` was loaded (`accsupp`) this will display the value of the `symbol` key with the accessibility support enabled for that key (`symbolaccess`). If there is no accessibility support, this just uses `\glsentrysymbol`.

The all caps version of `\glsaccesssymbolplural`.

The sentence case version of `\glsaccesssymbolplural`.
If accessibility support was enabled when `glossaries-extra` was loaded (`accsupp`) this will display the value of the `symbolplural` key with the accessibility support enabled for that key (`symbolpluralaccess`). If there is no accessibility support, this just uses \glsentrysymbolplural.

\GLSaccesstext{⟨entry-label⟩}

The all caps version of \glsaccesstext.

\Glsaccesstext{⟨entry-label⟩}

The sentence case version of \glsaccesstext.

\glssaccesstext{⟨entry-label⟩}

If accessibility support was enabled when `glossaries-extra` was loaded (`accsupp`) this will display the value of the `text` key with the accessibility support enabled for that key (`text-access`). If there is no accessibility support, this just uses \glsentrytext.

\GLSaccessuseri{⟨entry-label⟩}

The all caps version of \glsaccessuseri.

\Glsaccessuseri{⟨entry-label⟩}

The sentence case version of \glsaccessuseri.
If accessibility support was enabled when glossaries-extra was loaded (\texttt{accsupp}) this will display the value of the user1 key with the accessibility support enabled for that key (\texttt{user1-access}). If there is no accessibility support, this just uses \texttt{\textbackslash glsentryuseri}.

\begin{verbatim}
\texttt{\textbackslash GLSaccessuserii\{\langle entry-label\rangle\}}
\end{verbatim}

The all caps version of \texttt{\textbackslash glsaccessuserii}.

\begin{verbatim}
\texttt{\textbackslash Glsaccessuserii\{\langle entry-label\rangle\}}
\end{verbatim}

The sentence case version of \texttt{\textbackslash glsaccessuserii}.

\begin{verbatim}
\texttt{\textbackslash glsaccessuserii\{\langle entry-label\rangle\}}
\end{verbatim}

If accessibility support was enabled when glossaries-extra was loaded (\texttt{accsupp}) this will display the value of the user2 key with the accessibility support enabled for that key (\texttt{user2-access}). If there is no accessibility support, this just uses \texttt{\textbackslash glsentryuserii}.

\begin{verbatim}
\texttt{\textbackslash GLSaccessuseriii\{\langle entry-label\rangle\}}
\end{verbatim}

The all caps version of \texttt{\textbackslash glsaccessuseriii}.

\begin{verbatim}
\texttt{\textbackslash Glsaccessuseriii\{\langle entry-label\rangle\}}
\end{verbatim}

The sentence case version of \texttt{\textbackslash glsaccessuseriii}.

\begin{verbatim}
\texttt{\textbackslash glsaccessuseriii\{\langle entry-label\rangle\}}
\end{verbatim}

If accessibility support was enabled when glossaries-extra was loaded (\texttt{accsupp}) this will display the value of the user3 key with the accessibility support enabled for that key (\texttt{user3-access}). If there is no accessibility support, this just uses \texttt{\textbackslash glsentryuseriii}.

722
The all caps version of `\glsaccessuseriv`.

The sentence case version of `\glsaccessuseriv`.

If accessibility support was enabled when `glossaries-extra` was loaded (`accsupp`) this will display the value of the `user4` key with the accessibility support enabled for that key (`user4-access`). If there is no accessibility support, this just uses `\glsentryuseriv`.

The all caps version of `\glsaccessuserv`.

The sentence case version of `\glsaccessuserv`.

If accessibility support was enabled when `glossaries-extra` was loaded (`accsupp`) this will display the value of the `user5` key with the accessibility support enabled for that key (`user5-access`). If there is no accessibility support, this just uses `\glsentryuserv`.

The all caps version of `\glsaccessuservi`.
The sentence case version of `\glsaccessuservi`.

If accessibility support was enabled when `glossaries-extra` was loaded (`accsupp`) this will display the value of the `user6` key with the accessibility support enabled for that key (`user6-access`). If there is no accessibility support, this just uses `\glsentryuservi`.

`\glsaccsupp{⟨replacement⟩}{⟨content⟩}`

Applies `⟨replacement⟩` as the ActualText for `⟨content⟩` using `\glsaccessibility`.

`\glsacspace{⟨label⟩}`

Uses a non-breakable space if the short form is less than `\glsacspacemax` otherwise uses `\space`. This command is provided by `glossaries` but has a hard-coded maximum of 3em. This command is redefined by `glossaries-extra` to use `\glsacspacemax`.

`\glsacspacemax`

Expands to the maximum value used by `\glsacspace`. This is a macro not a register. The default is 3em.

`\glsadd[⟨options⟩]{⟨entry-label⟩}`

Indexes the entry identified by `⟨entry-label⟩`.

`\glsaddall[⟨options⟩]`

Indexes all entries.
Iterates over all glossaries (or all those listed in the \texttt{types} option) and indexes each entry in the glossary. The optional argument \texttt{⟨options⟩} are passed to \texttt{\glsadd}. This command can't be used with \texttt{bib2gls}. Use the \texttt{selection=all} resource option instead.

\begin{verbatim}
\glsaddallunindexed[⟨glossary types⟩]
glossaries-extra v1.49+
\end{verbatim}

Iterates over all glossaries listed in \texttt{⟨glossary types⟩} and indexes each entry (with format=\texttt{glsignore}) that hasn't already been indexed. This command can't be used with \texttt{bib2gls}. Use the \texttt{selection=all} resource option instead.

\begin{verbatim}
\glsaddallunused[⟨glossary types⟩]
glossaries v3.08a+
\end{verbatim}

Iterates over all glossaries listed in \texttt{⟨glossary types⟩} and indexes each entry (with format=\texttt{glsignore}) that hasn't been used. This command can't be used with \texttt{bib2gls}. Use the \texttt{selection=all} resource option instead.

\begin{verbatim}
\glsaddeach[⟨options⟩] {⟨entry label list⟩}
glossaries-extra v1.31+
\end{verbatim}

Does \texttt{\glsadd[⟨options⟩] {⟨entry-label⟩}} for each label in the supplied comma-separated list.

\begin{verbatim}
\glsadddkey{⟨key⟩}{⟨default value⟩}{⟨no link cs⟩}{⟨no link ucfirst cs⟩}{⟨link cs⟩}{⟨link ucfirst cs⟩}{⟨link allcaps cs⟩}
glossaries v3.12a
\end{verbatim}

 Defines a new glossary entry key with the given default value and commands that are analogous to \texttt{\glsentrytext{⟨no link cs⟩}}, \texttt{\Glsentrytext{⟨no link ucfirst cs⟩}}, \texttt{\glstext{⟨link cs⟩}}, \texttt{\Glstext{⟨link ucfirst cs⟩}}, \texttt{\GLStext{⟨link allcaps cs⟩}}. The starred version switches on field expansion for the given key.

\begin{verbatim}
\glsaddpostsetkeys
glossaries-extra v1.30+
\end{verbatim}

Hook implemented after setting the options passed to \texttt{\glsadd}.

\begin{verbatim}
\glsaddpresetkeys
glossaries-extra v1.30+
\end{verbatim}
Hook implemented before setting the options passed to \glsadd.

\glsaddstoragekey\{\langle key \rangle \}\{\langle default value \rangle \}\{\langle no link cs \rangle \}
glossaries v4.16

Provides a new glossary entry key with a default value and a command for simply accessing the value (without indexing or hyperlinks). The starred version switches on field expansion for the given key.

\glsaltlistitem\{\langle entry-label \rangle \}
glossaries-extra-stylemods v1.47+

Used to display the top-level entry item in the altlist styles.

\glsalttreechildpredesc
glossaries-extra-stylemods v1.46+

Inserted before the child descriptions for the alttree styles.

\glsalttreepredesc
glossaries-extra-stylemods v1.46+

Inserted before the top-level descriptions for the alttree styles.

\glsapptopostlink\{\langle category \rangle \}\{\langle code \rangle \}
glossaries-extra v1.49+

§5.5.4; 251

Appends \langle code \rangle to post-link hook associated with the category identified by the label \langle category \rangle (or simply defines it, if it doesn’t already exist).

\glsautoprefix
glossaries v1.14+

Expands to the prefix for the label used by numberedsection=autolabel and numberedsection=nameref.

\glsbackslash
glossaries v4.11+

726
Expands to a literal backslash.

\glscapitalisewords{⟨content⟩}
glossaries v4.48+

§5.2.4; 200

Just does \capitalisewords but may be redefined to use \capitalisefmtwords, if required.

\glscapscase{⟨no change⟩}{⟨sentence⟩}{⟨all caps⟩}
glossaries

Initialised by the \gls-like and \glstext-like commands, this expands to ⟨no change⟩ if the calling command doesn’t apply a case-change (such as \gls or \glstext), to ⟨sentence⟩ if the calling command converts to sentence case (such as \Gls or \Glstext), or to ⟨all caps⟩ if the calling command converts to all caps (such as \GLS or \GLStext). This command may be used within associated hooks, entry display styles (\defglsentryfmt), and the post-link hook.

\glscapturedgroup{n}
glossaries-extra-bib2gls v1.31+

Expands to \string\$⟨n⟩. Note that this isn’t the same as \MGP.

\glscategory{⟨entry-label⟩}

§10; 517

Expands to the entry’s category.

\glscategorylabel

§4.5.3.1; 163

Expands to the category label of the abbreviation that is in the process of being defined by \newabbreviation. Maybe used in the style hooks (but take care to expand this command, if necessary).

\glscombinedfirstsep{⟨prev label⟩}{⟨next label⟩}
glossaries-extra v1.48+

§7.4; 346

727
Separator used between elements of a multi-entry set where only the next element have been marked as used.

\glscombinedfirstsepfirst\{\langle prev label\rangle\}\{\langle next label\rangle\} \quad \text{glossaries-extra v1.48+} \quad \S 7.4; 347

Separator used between elements of a multi-entry set where neither the previous nor the next element has been marked as used.

\glscombinedsep\{\langle prev label\rangle\}\{\langle next label\rangle\} \quad \text{glossaries-extra v1.48+} \quad \S 7.4; 346

Separator used between elements of a multi-entry set where both elements have been marked as used.

\glscombinedsepfirst\{\langle prev label\rangle\}\{\langle next label\rangle\} \quad \text{glossaries-extra v1.48+} \quad \S 7.4; 347

Separator used between elements of a multi-entry set where only the previous element have been marked as used.

\glscurrententrylabel \quad \text{glossaries v3.02+}

Assigned at the start of each entry item within the glossary. This command may be used by glossary hooks, such as the post-description hook, to reference the current entry.

\glscurrententrylevel \quad \text{glossaries-extra v1.44+} \quad \S 8.4.3; 405

Defined within the “unsrt” family of commands to the current hierarchical level (taking leveloffset into account).

\glscurrentfieldvalue \quad \text{glossaries v4.23+}

Conditional commands such as \ifgls_hasfield set this to the field’s value for use within the \langle true\rangle code.
Command Summary

£\textbackslash{}glscurrentrootentry£ glossaries-extra v1.49+

May be used within £\textbackslash{}printunsrtglossaryentryprocesshook£ to reference the most recent top level entry label (allowing for flatten but not leveloffset).

£\textbackslash{}glscurrenttoplevelentry£ glossaries-extra v1.49+

May be used within £\textbackslash{}printunsrtglossaryentryprocesshook£ to reference the most recent top level entry label (allowing for flatten and leveloffset).

£\textbackslash{}glscustomtext£ glossaries

The custom text provided by £\textbackslash{}glsdisp£ or the link text for the £\textbackslash{}glstext£-like commands. This command may be used within associated hooks, entry display styles (£\textbackslash{}defglsentryfmt£), and the post-link hook.

£\textbackslash{}glsdefaultshortaccess\{\langle long\rangle\}\{\langle short\rangle\}£ glossaries-accsupp v4.45+ (requires accsupp)

Used when £newabbreviation£ automatically assigns shortaccess. This is defined by glossaries-accsupp to just do £\langle long\rangle£ but is redefined by glossaries-extra to do £\langle long\rangle£ (£\langle short\rangle£).

£\textbackslash{}glsdefaulttype£ initial: main glossaries

Expands to the label of the default glossary, which is normally main but if nomain is used, it will be the label of the first glossary to be defined.

£\textbackslash{}glsdefpostdesc\{\langle category\rangle\}\{\langle definition\rangle\}£ glossaries-extra v1.31+

§8.6.2; 428

Defines post-description hook associated with the category identified by the label £\langle category\rangle£. This simply (re)defines £\textbackslash{}glsxtrpostdesc\langle category\rangle£ for the given £\langle category\rangle£ to £\langle definition\rangle£.

£\textbackslash{}glsdefpostlink\{\langle category\rangle\}\{\langle definition\rangle\}£ glossaries-extra v1.31+

§5.5.4; 251
Command Summary

Defines post-link hook associated with the category identified by the label \langle category \rangle. This simply (re)defines \glsxtrpostlink\langle category \rangle for the given \langle category \rangle to \langle definition \rangle.

\texttt{\glsdefpostname\{\langle category \rangle\}\{\langle definition \rangle\}} \quad \text{glossaries-extra v1.31+}

Defines post-name hook associated with the category identified by the label \langle category \rangle. This simply (re)defines \glsxtrpostname\langle category \rangle for the given \langle category \rangle to \langle definition \rangle.

\texttt{\GLSdesc\[\langle options \rangle\]\{\langle entry-label \rangle\}\[\langle insert \rangle\]} \quad \text{modifiers: } * + \langle alt-mod \rangle \quad \text{glossaries}

As \texttt{\glsdesc} but converts the link text to all caps.

\texttt{\Glsdesc\[\langle options \rangle\]\{\langle entry-label \rangle\}\[\langle insert \rangle\]} \quad \text{modifiers: } * + \langle alt-mod \rangle \quad \text{glossaries}

As \texttt{\glsdesc} but converts the link text to sentence case. Use \texttt{\Glossentrydesc} within custom glossary styles instead of this command.

\texttt{\glsdesc\[\langle options \rangle\]\{\langle entry-label \rangle\}\[\langle insert \rangle\]} \quad \text{modifiers: } * + \langle alt-mod \rangle \quad \text{glossaries}

References the entry identified by \langle entry-label \rangle. The text produced is obtained from the description value. The \langle insert \rangle argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \texttt{\glslink} options. Use \texttt{\glossentrydesc} within custom glossary styles instead of this command.

\texttt{\glssdescplural\[\langle options \rangle\]\{\langle entry-label \rangle\}\[\langle insert \rangle\]} \quad \text{modifiers: } * + \langle alt-mod \rangle \quad \text{glossaries}

As \texttt{\glsdesc} but for the descriptionplural field.

\texttt{\glsdescriptionaccessdisplay\{\langle text \rangle\}\{\langle entry-label \rangle\}} \quad \text{glossaries-accsupp}

§8.6.1; 426
**Command Summary**

Does *(text)* with the *descriptionaccess* replacement text (if set).

\glsdescriptionpluralaccessdisplay\{⟨text⟩\}\{⟨entry-label⟩\}  glossaries-accsupp

Does *(text)* with the *descriptionpluralaccess* replacement text (if set).

\glsdescwidth  glossary-long & glossary-super

A length register used to set the width of the description column for tabular-like styles.

\Glsdisp[\langle options\rangle]\{⟨entry-label⟩\}\{⟨text⟩\}  modifiers: * + ⟨alt-mod⟩  glossaries v4.50+

As \glsdisp but sets the link text to *\glssentencecase{⟨text⟩}*. This is provided to allow a sentence case mapping in the event that \glsdisp occurs at the start of content that has automated case-changing.

\glsdisp[\langle options\rangle]\{⟨entry-label⟩\}\{⟨text⟩\}  modifiers: * + ⟨alt-mod⟩  glossaries v1.19+

References the entry identified by ⟨*entry-label*⟩ with the given ⟨*text*⟩ as the link text. This command unsets the first use flag (use \glslink instead, if the first use flag should not be altered). This command is considered a \gls-like command. For the first optional argument, see \glslink options.

\glsdisplaynumberlist\{⟨entry-label⟩\}  glossaries v3.02+  §11.5.8; 597

Formats the location list for the given entry. Redefined by glossaries-extra-bib2gls to obtain the location list from the location field.

\glsdoifexists\{⟨entry-label⟩\}\{⟨code⟩\}  glossaries
Command Summary

Does \textit{\langle code\rangle} if the entry given by \textit{\langle entry-label\rangle} exists. If the entry doesn’t exist, this will either generate an error (\texttt{undefaction=error}) or a warning (\texttt{undefaction=warn}) and, within the document environment, it will insert the unknown marker ??.

\begin{verbatim}
\glsdoifexistsordo{\langle entry-label\rangle}{\langle true\rangle}{\langle false\rangle} \text{ glossaries v4.19+}
\end{verbatim}

Similar to \texttt{\ifglsentryexists}, this does \textit{\langle true\rangle} if the entry given by \textit{\langle entry-label\rangle} exists. If the entry doesn’t exist, this does \textit{\langle false\rangle} and generates an error (\texttt{undefaction=error}) or a warning (\texttt{undefaction=warn}). The unknown marker ?? will be placed before the \textit{\langle false\rangle} code.

\begin{verbatim}
\glsdoifexistsorwarn{\langle entry-label\rangle}{\langle code\rangle} \text{ glossaries v4.03+}
\end{verbatim}

Like \texttt{\glsdoifexists}, but always warns (no error) if the entry doesn’t exist, regardless of the \texttt{undefaction} setting, and doesn’t show the unknown marker.

\begin{verbatim}
\glsdoifnoexists{\langle entry-label\rangle}{\langle code\rangle} \text{ glossaries}
\end{verbatim}

Does \textit{\langle code\rangle} if the entry given by \textit{\langle entry-label\rangle} does not exist. If the entry does exist, this will either generate an error (\texttt{undefaction=error}) or a warning (\texttt{undefaction=warn}).

\begin{verbatim}
\glstenableentrycount \text{ glossaries v4.14+}
\end{verbatim}

Enables entry counting.

\begin{verbatim}
\glstenableentryunitcount \text{ §6.1; 316}
\end{verbatim}

Enables entry unit counting.

\begin{verbatim}
\glsencapwrcontent{\langle code\rangle} \text{ glossaries v4.50+ & glossaries-extra v1.49+}
\end{verbatim}

Encapsulates the indexing code (within \texttt{\glslinkwrcontent}).
\texttt{\textbackslash glsendrange[\langle options\rangle]\{\langle entry label list\rangle\}} \hspace{1cm} \texttt{glossaries-extra v1.50+} \\
\noindent As \texttt{\textbackslash glosstartrange} but with the end range marker \texttt{)}. \\

\texttt{\textbackslash glsentrycounterlabel} \hspace{1cm} \texttt{glossaries v3.0+} \\
\noindent Used by \texttt{\textbackslash glosentryitem} to display the entry counter label. \\

\texttt{\textbackslash glsentrycurrcount\{\langle entry-label\rangle\}} \hspace{1cm} \texttt{glossaries v4.14+} \\
\noindent Expands to the current entry count running total or 0 if not available (needs to be enabled with \texttt{\textbackslash glosenableentrycount} or \texttt{\textbackslash glosenableentryunitcount}). With unit entry counting, this expands to the total for the current unit. \\

\texttt{\Glsentrydesc\{\langle entry-label\rangle\}} \hspace{1cm} \texttt{glossaries} \\
\noindent Partially robust command that displays the value of the description field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command. \\

\texttt{\textbackslash glsentrydesc\{\langle entry-label\rangle\}} \hspace{1cm} \texttt{glossaries} \\
\noindent Simply expands to the value of the description field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the description field doesn’t contain any fragile commands. \\

\texttt{\textbackslash glsentrydescplural\{\langle entry-label\rangle\}} \hspace{1cm} \texttt{glossaries} \\
\noindent Simply expands to the value of the descriptionplural field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the descriptionplural field doesn’t contain any fragile commands.
\glsentryfirst{⟨entry-label⟩}  \textit{glossaries}

Partially robust command that displays the value of the \texttt{first} field with the first letter converted to uppercase. As from \texttt{glossaries} v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\texttt{\glsentryfirst{⟨entry-label⟩}}  \textit{glossaries}

Simply expands to the value of the \texttt{first} field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the \texttt{first} field doesn’t contain any fragile commands.

\glsentryfirstplural{⟨entry-label⟩}  \textit{glossaries}

Partially robust command that displays the value of the \texttt{firstplural} field with sentence case applied. As from \texttt{glossaries} v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\texttt{\glsentryfirstplural{⟨entry-label⟩}}  \textit{glossaries}

Simply expands to the value of the \texttt{firstplural} field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the \texttt{firstplural} field doesn’t contain any fragile commands.

\texttt{\glsentryfmt}  \textit{glossaries v3.11a+}

The default display format used by the \texttt{\gls} like commands. This checks if the \texttt{short} field has been set for the current entry and, if set, initialises the abbreviation formatting commands (with \texttt{\glssetabbrvfmt}). This command will do \texttt{\glsgenentryfmt} (encapsulated with \texttt{\glsxtrregularfont}) if the entry is considered a regular entry (\texttt{\glsifregular}) or if the entry doesn’t have the \texttt{short} field set. Otherwise it will do \texttt{\glsxtrgenabbrvfmt} encapsulated with \texttt{\glsxtrabbreviationfont}.

\texttt{\glsentryindexcount{⟨entry-label⟩}}  \textit{glossaries-extra v1.49+}  \textit{§5.8; 268}

734
Expands to the number of times the given entry has been indexed. This will expand to 0 if the entry hasn’t been indexed or hasn’t been defined.

\texttt{\textbackslash glsentryitem\{entry-label\}} \hspace{1cm} \textit{glossaries v3.0+}

Does nothing if \texttt{entrycounter=false}, otherwise increments and displays the associated counter.

\texttt{\textbackslash Glsentrylong\{entry-label\}} \hspace{1cm} \textit{glossaries v3.0+}

Displays the value of the \texttt{long} field with sentence case applied. Does nothing if the entry hasn’t been defined. As from \textit{glossaries v4.50}, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\texttt{\textbackslash glsentrylong\{entry-label\}} \hspace{1cm} \textit{glossaries v3.0+}

Simply expands to the value of the \texttt{long} field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the \texttt{long} field doesn’t contain any fragile commands.

\texttt{\textbackslash Glsentrylongpl\{entry-label\}} \hspace{1cm} \textit{glossaries v3.0+}

Displays the value of the \texttt{longplural} field with sentence case applied. Does nothing if the entry hasn’t been defined. As from \textit{glossaries v4.50}, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\texttt{\textbackslash glsentrylongpl\{entry-label\}} \hspace{1cm} \textit{glossaries v3.0+}

Simply expands to the value of the \texttt{longplural} field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the \texttt{longplural} field doesn’t contain any fragile commands.

\texttt{\textbackslash Glsentryname\{entry-label\}} \hspace{1cm} \textit{glossaries}

735
\glsentryname{(entry-label)} \hspace{5cm} \text{glossaries}

Simply expands to the value of the \texttt{name} key. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the \texttt{name} key doesn’t contain any fragile commands.

\glsentrynumberlist{(entry-label)} \hspace{5cm} \text{glossaries v3.02+} \hspace{2cm} \text{§11.5.8; 598}

Displays the location list for the given entry. Redefined by \texttt{glossaries-extra-bib2gls} to obtain the location list from the \texttt{location} field.

\glsentryparent{(entry-label)} \hspace{5cm} \text{glossaries}

Expands to the value of the \texttt{parent} field. Expands to nothing if the \texttt{parent} field hasn’t been set and expands to \texttt{\relax} if the entry hasn’t been defined.

\glsentrypdfsymbol{(entry-label)} \hspace{5cm} \text{glossaries-extra v1.42+} \hspace{2cm} \text{§8.6; 424}

Used when \texttt{\glossentrysymbol} occurs in a PDF bookmark.

\Glsentryplural{(entry-label)} \hspace{5cm} \text{glossaries}

Partially robust command that displays the value of the \texttt{plural} field with sentence case applied. As from \texttt{glossaries v4.50}, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryplural{(entry-label)} \hspace{5cm} \text{glossaries}
Command Summary

Simply expands to the value of the \texttt{plural} field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the \texttt{plural} field doesn’t contain any fragile commands.

\texttt{\textbackslash glsentryprevcount\{\langle entry-label\rangle\}} \hspace{1cm} \text{glossaries v4.14+}

Expands to the final entry count total from the previous \LaTeX{} run or if 0 if not available (needs to be enabled with \texttt{\textbackslash glsenableentrycount} or \texttt{\textbackslash glsenableentryunitcount}). With unit entry counting, this expands to the total for the current unit.

\texttt{\textbackslash glsentryprevmaxcount\{\langle entry-label\rangle\}} \hspace{1cm} \text{glossaries v4.14+}

§6.1; 317

Expands to the maximum entry unit count total from the previous \LaTeX{} run or if 0 if not available (needs to be enabled with \texttt{\textbackslash glsenableentryunitcount}).

\texttt{\textbackslash glsentryprevtotalcount\{\langle entry-label\rangle\}} \hspace{1cm} \text{glossaries v4.14+}

§6.1; 317

Expands to the final entry count total from the previous \LaTeX{} run or if 0 if not available (needs to be enabled with \texttt{\textbackslash glsenableentryunitcount}).

\texttt{\textbackslash Glsentryshort\{\langle entry-label\rangle\}} \hspace{1cm} \text{glossaries v3.0+}

Displays the value of the \texttt{short} field with sentence case applied. Does nothing if the entry hasn’t been defined. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\texttt{\textbackslash Glsentryshort\{\langle entry-label\rangle\}} \hspace{1cm} \text{glossaries v3.0+}

Simply expands to the value of the \texttt{short} field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the \texttt{short} field doesn’t contain any fragile commands.

\texttt{\textbackslash Glsentryshortpl\{\langle entry-label\rangle\}} \hspace{1cm} \text{glossaries v3.0+}
Command Summary

Displays the value of the shortplural field with sentence case applied. Does nothing if the entry hasn’t been defined. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryshortpl\{⟨entry-label⟩\}  
glossaries v3.0+

Simply expands to the value of the shortplural field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the shortplural field doesn’t contain any fragile commands.

\Glsentriesymbol\{⟨entry-label⟩\}  
glossaries

Partially robust command that displays the value of the symbol field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentriesymbol\{⟨entry-label⟩\}  
glossaries

Simply expands to the value of the symbol field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the symbol field doesn’t contain any fragile commands.

\glsentriesymbolaccess\{⟨entry-label⟩\}  
glossaries–accsupp

As \glsentriesymbol but for the symbolaccess field.

\glsentriesymbolplural\{⟨entry-label⟩\}  
glossaries

Simply expands to the value of the symbolplural field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the symbolplural field doesn’t contain any fragile commands.

\Glsentrytext\{⟨entry-label⟩\}  
glossaries

738
Command Summary

Partially robust command that displays the value of the text field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentrytext{⟨entry-label⟩}
glossaries

Simply expands to the value of the text field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the text field doesn’t contain any fragile commands.

\glsentrytitlecase{⟨entry-label⟩}⟨(field-label)⟩
glossaries v4.22+

Applies title case to the value supplied in the given field (which is obtained with \glsfield-fetch).

\glsentrytype{⟨entry-label⟩}
glossaries

Simply expands to the value of the type key. Does nothing if the entry hasn’t been defined.

\glsentryuseri{⟨entry-label⟩}
glossaries v2.04+

Partially robust command that displays the value of the user1 field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuseri{⟨entry-label⟩}
glossaries v2.04+

Simply expands to the value of the user1 field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the user1 field doesn’t contain any fragile commands.

\glsentryuserii{⟨entry-label⟩}
glossaries v2.04+

Simply expands to the value of the user1 field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the user1 field doesn’t contain any fragile commands.
Command Summary

Partially robust command that displays the value of the user2 field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuserii\{⟨entry-label⟩\}
glossaries v2.04+

Simply expands to the value of the user2 field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the user2 field doesn’t contain any fragile commands.

\Glsentryuserii\{⟨entry-label⟩\}
glossaries v2.04+

Partially robust command that displays the value of the user3 field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuseriii\{⟨entry-label⟩\}
glossaries v2.04+

Simply expands to the value of the user3 field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the user3 field doesn’t contain any fragile commands.

\Glsentryuseriii\{⟨entry-label⟩\}
glossaries v2.04+

Partially robust command that displays the value of the user4 field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuseriv\{⟨entry-label⟩\}
glossaries v2.04+

Simply expands to the value of the user4 field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the user4 field doesn’t contain any fragile commands.
Command Summary

\Glsentryuserv\{\langle entry-label \rangle \}\{\text{glossaries v2.04+}\}

Partially robust command that displays the value of the user5 field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuserv\{\langle entry-label \rangle \}\{\text{glossaries v2.04+}\}

Simply expands to the value of the user5 field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the user5 field doesn’t contain any fragile commands.

\Glsentryuservi\{\langle entry-label \rangle \}\{\text{glossaries v2.04+}\}

Partially robust command that displays the value of the user6 field with sentence case applied. As from glossaries v4.50, this command can expand in PDF bookmarks. Outside of PDF bookmarks it will expand to a robust internal command.

\glsentryuservi\{\langle entry-label \rangle \}\{\text{glossaries v2.04+}\}

Simply expands to the value of the user6 field. Does nothing if the entry hasn’t been defined. May be used in expandable contexts provided that the user6 field doesn’t contain any fragile commands.

\glsexclapplyinnerfmtfield\{\langle entry-label \rangle \}\{\langle internal-field \rangle \}\{\text{glossaries-extra v1.49+}\}

Locally adds the field given by its internal field label \langle internal-field \rangle to the inner formatting exclusion list for the entry identified by \langle entry-label \rangle. This typically means that the field value already contains the inner formatting.

\glsexpandfields\{\text{glossaries v3.08a+}\}

§5.5.3; 246
Expand field values when defining entries, except for those that explicitly have expansion disabled with \glssetnoexpandfield.

\glsextrapostnamehook{⟨entry-label⟩}
glossaries-extra v1.25+

A general purpose hook that’s performed within \glsxtrpostnamehook.

\glsfieldaccsupp{⟨replacement⟩}{⟨content⟩}{⟨field⟩}{⟨entry-label⟩}
glossaries-accsupp v4.45+

If glossaries-extra has been loaded, this command will first check for the existence of the command \glsxtr{(category)}{(field)}accsupp. If that command doesn’t exist or if glossaries-extra hasn’t been loaded, it then checks for the existence of \gls{(field)}accsupp (for example, \glsshortaccsupp). Failing that it will use \glsaccsupp. Whichever command is found first, ⟨cs⟩{⟨replacement⟩}{⟨content⟩} is performed.

\glsfielddef{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}
glossaries v4.16+

Locally assigns the ⟨value⟩ to the given field (identified by the internal field label ⟨field⟩) for the entry identified by ⟨entry-label⟩. Produces an error (or warning with undefaction=warn) if the entry or field doesn’t exist. Note that this doesn’t update any associated fields.

\glsfieldedef{⟨entry-label⟩}{⟨field⟩}{⟨value⟩}
glossaries v4.16+

Locally assigns the full expansion of ⟨value⟩ to the given field (identified by the internal field label ⟨field⟩) for the entry identified by ⟨entry-label⟩. Produces an error (or warning with undefaction=warn) if the entry or field doesn’t exist. Note that this doesn’t update any associated fields.

\glsfieldfetch{⟨entry-label⟩}{⟨field-label⟩}{⟨cs⟩}
glossaries v4.16+

Fetches the value of the given field for the given entry and stores it in the command ⟨cs⟩. Triggers an error if the given field (identified by its internal field label) hasn’t been defined. Uses \glsdoifexists.
Command Summary

\glsfieldgdef\{\langle\text{entry-label}\rangle}\{\langle\text{field}\rangle}\{\langle\text{value}\rangle}\} \quad \text{glossaries v4.16+}

As \glsfielddef but does a global assignment.

\glsfieldxdef\{\langle\text{entry-label}\rangle}\{\langle\text{field}\rangle}\{\langle\text{value}\rangle}\} \quad \text{glossaries v4.16+}

As \glsfieldedef but does a global assignment.

\glsFindWidestAnyName\[\langle\text{glossary labels}\rangle]\] \quad \text{glossaries-extra-stylemods v1.05+}

§8.6.5.4; 441

Finds and sets the widest name for all entries in the given glossaries.

\glsFindWidestAnyNameLocation\[\langle\text{glossary labels}\rangle\}\{\langle\text{register}\rangle}\] \quad \text{glossaries-extra-stylemods v1.05+}

§8.6.5.4; 442

Like \glsFindWidestAnyName but also also measures the location list. The length of the widest location is stored in \(\langle\text{register}\rangle\), which should be a length register.

\glsFindWidestAnyNameSymbol\[\langle\text{glossary labels}\rangle\}\{\langle\text{register}\rangle\] \quad \text{glossaries-extra-stylemods v1.05+}

§8.6.5.4; 441

Like \glsFindWidestAnyName but also also measures the symbol. The length of the widest symbol is stored in \(\langle\text{register}\rangle\) which should be a length register.

\glsFindWidestAnyNameSymbolLocation\[\langle\text{glossary labels}\rangle\]\{\langle\text{register1}\rangle\}\{\langle\text{register2}\rangle\} \quad \text{glossaries-extra-stylemods v1.05+}

§8.6.5.4; 442

Like \glsFindWidestAnyNameSymbol but also also measures the location list. The length of the widest symbol is stored in \(\langle\text{register1}\rangle\) and the length of the widest location is stored in \(\langle\text{register2}\rangle\), which should both be length registers.

\glsFindWidestLevelTwo\[\langle\text{glossary labels}\rangle\] \quad \text{glossaries-extra-stylemods v1.05+}

§8.6.5.4; 441
Finds and sets the widest name for all entries with hierarchical level less than or equal to 2 in the given glossaries.

\texttt{\textbackslash glsFindWidestTopLevelName} \hspace{1cm} \text{glossaries-extra-stylemods} \hspace{1cm} \S 8.6.5.4; 440

A synonym for \texttt{\textbackslash glsfindwidesttoplevelname}.

\texttt{\textbackslash glsfindwidesttoplevelname}\[\langle\text{glossary labels}\rangle\] \hspace{1cm} \text{glossary-tree v4.22+} \hspace{1cm} \S 8.6.5.4; 441

Finds and sets the widest name for all top-level entries in the given glossaries. If the optional argument is omitted, the list of all non-ignored glossaries is assumed.

\texttt{\textbackslash glsFindWidestUsedAnyName}\[\langle\text{glossary labels}\rangle\] \hspace{1cm} \text{glossaries-extra-stylemods v1.05+} \hspace{1cm} \S 8.6.5.4; 442

Finds and sets the widest name for all entries that have been marked as used in the given glossaries.

\texttt{\textbackslash glsFindWidestUsedAnyNameLocation}\[\langle\text{glossary labels}\rangle]\{\langle\text{register}\rangle\} \hspace{1cm} \text{glossaries-extra-stylemods v1.05+} \hspace{1cm} \S 8.6.5.4; 442

Like \texttt{\textbackslash glsFindWidestUsedAnyName} but also also measures the location list. The length of the widest location is stored in \langle\text{register}\rangle, which should be a length register.

\texttt{\textbackslash glsFindWidestUsedAnyNameSymbol}\[\langle\text{glossary labels}\rangle]\{\langle\text{register}\rangle\} \hspace{1cm} \text{glossaries-extra-stylemods v1.05+} \hspace{1cm} \S 8.6.5.4; 441

Like \texttt{\textbackslash glsFindWidestUsedAnyName} but also also measures the symbol. The length of the widest symbol is stored in \langle\text{register}\rangle which should be a length register.

\texttt{\textbackslash glsFindWidestUsedAnyNameSymbolLocation}\[\langle\text{glossary labels}\rangle]\{\langle\text{register1}\rangle\}\{\langle\text{register2}\rangle\} \hspace{1cm} \text{glossaries-extra-stylemods v1.05+} \hspace{1cm} \S 8.6.5.4; 442
Command Summary

Like \texttt{\textbackslash glsFindWidestUsedAnyNameSymbol} but also also measures the location list. The length of the widest symbol is stored in \texttt{\langle register1 \rangle} and the length of the widest location is stored in \texttt{\langle register2 \rangle}, which should both be length registers.

\texttt{\textbackslash glsFindWidestUsedLevelTwo[\langle glossary\ labels\ \rangle]} \hspace{1cm} \textit{glossaries-extra-stylemods v1.05+} \hspace{1cm} \texttt{§8.6.5.4; 441}

Finds and sets the widest name for all entries that have been marked as used with hierarchical level less than or equal to 2 in the given glossaries.

\texttt{\textbackslash glsFindWidestUsedTopLevelName[\langle glossary\ labels\ \rangle]} \hspace{1cm} \textit{glossaries-extra-stylemods v1.05+} \hspace{1cm} \texttt{§8.6.5.4; 441}

Finds and sets the widest name for all top-level entries that have been marked as used in the given glossaries.

\texttt{\textbackslash GLSfirst[\langle options\ \rangle]{\langle entry-label\ \rangle}{\langle insert\ \rangle} modifiers: * + \langle alt-mod\ \rangle} \hspace{1cm} \textit{glossaries}

As \texttt{\textbackslash glsfirst} but converts the link text to all caps. If you have defined the entry with \texttt{\textbackslash newabbreviation use \textbackslash GLSxtrfull} or \texttt{\textbackslash GLS[prereset]} instead.

\texttt{\textbackslash Glsfirst[\langle options\ \rangle]{\langle entry-label\ \rangle}{\langle insert\ \rangle} modifiers: * + \langle alt-mod\ \rangle} \hspace{1cm} \textit{glossaries}

As \texttt{\textbackslash glsfirst} but converts the first character of the link text to uppercase (for the start of a sentence) using \texttt{\textbackslash makefirstuc}. If you have defined the entry with \texttt{\textbackslash newabbreviation use \textbackslash GLsxtrfull} or \texttt{\textbackslash Gls[prereset]} instead.

\texttt{\textbackslash glsfirst[\langle options\ \rangle]{\langle entry-label\ \rangle}{\langle insert\ \rangle} modifiers: * + \langle alt-mod\ \rangle} \hspace{1cm} \textit{glossaries}

References the entry identified by \texttt{\langle entry-label \rangle}. The text produced is obtained from the \texttt{first} value. The \texttt{\langle insert \rangle} argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. If you have defined the entry with \texttt{\textbackslash newabbreviation use \textbackslash glsxtrfull} for the full form or \texttt{\textbackslash glsxtrlong} for the long form or use \texttt{\textbackslash gls[prereset]}, as some abbreviation styles are too complicated to work with \texttt{\textbackslash glsfirst}. For the first optional argument, see \texttt{\textbackslash glslink options}.
Command Summary

\texttt{\textbackslash glsfir\text{tabbrvdefaultfont}\{\langle text\rangle\}}

Formatting command for the short form on first use used by the abbreviation styles that don’t apply a font change by default.

\texttt{\textbackslash glsfir\text{tabbrvemfont}\{\langle text\rangle\}}

Short form font used by the “em” abbreviation styles on first use.

\texttt{\textbackslash glsfir\text{tabbrvfont}\{\langle text\rangle\}}

Font formatting command for the short form on first use, initialised by the abbreviation style.

\texttt{\textbackslash glsfir\text{tabbrvhyphenfont}\{\langle text\rangle\}}

Short form font used by the “hyphen” abbreviation styles on first use.

\texttt{\textbackslash glsfir\text{tabbrvonlyfont}\{\langle text\rangle\}}

Short form font used by the “only” abbreviation styles on first use.

\texttt{\textbackslash glsfir\text{tabbrvscfont}\{\langle text\rangle\}}

Short form font used by the small caps “sc” abbreviation styles on first use.

\texttt{\textbackslash glsfir\text{tabbrvsconlyfont}\{\langle text\rangle\}}

Short form font used by the “sc-only” abbreviation styles on first use.

\texttt{\textbackslash glsfir\text{tabbrvscuserfont}\{\langle text\rangle\}}

Short form font used by the “sc-only” abbreviation styles on first use.
Command Summary

Short form font used by the small caps “sc-user” abbreviation styles on first use.

\glsfirstabbrvsmfont{⟨text⟩}
glossaries-extra v1.17+

157

Short form font used by the “sm” abbreviation styles on first use.

\glsfirstabbrvuserfont{⟨text⟩}
glossaries-extra v1.04+

136

Short form font used by the “user” abbreviation styles on first use.

\glsfirstaccessdisplay{⟨text⟩}{⟨entry-label⟩}
glossaries-accsupp

Does ⟨text⟩ with the firstaccess replacement text (if set).

\glsfirstinnerfmtabbrvfont{⟨text⟩}
glossaries-extra v1.49+

§4.5.3.1; 166

Applies both \glsfirstabbrvfont and \glsxtrgenentrytextfmt to ⟨text⟩.

\glsfirstinnerfmtlongfont{⟨text⟩}
glossaries-extra v1.49+

§4.5.3.1; 168

Applies both \glsfirstlongfont and \glsxtrgenentrytextfmt to ⟨text⟩.

\glsfirstlongdefaultfont{⟨text⟩}

Formatting command for the long form on first use used by the abbreviation styles that don’t apply a font change by default.

\glsfirstlongemfont{⟨text⟩}
glossaries-extra v1.04+

158

Long form font used by the “em” abbreviation styles on first use.
\glsfirstlongfont{(text)}

Font formatting command for the long form on first use, initialised by the abbreviation style.

\glsfirstlongfootnotefont{(text)}\quad glossaries-extra v1.05+

Formatting command for the first use long form used by the footnote abbreviation styles.

\glsfirstlonghyphenfont{(text)}\quad glossaries-extra v1.17+

Long form font used by the “hyphen” abbreviation styles on first use.

\glsfirstlongonlyfont{(text)}\quad glossaries-extra v1.17+

Long form font used by the “only” abbreviation styles on first use.

\glsfirstlonguserfont{(text)}\quad glossaries-extra v1.04+

Long form font used by the “user” abbreviation styles on first use.

\GLSfirstplural[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

glossaries

As \glsfirstplural but converts the link text to all caps. If you have defined the entry with \newabbreviation use \GLSxtrfullpl or \GLSpl[prereset] instead.

\Glsfirstplural[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩

glossaries

As \glsfirstplural but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc. If you have defined the entry with \newabbreviation use \Glsxtrfullpl or \Glspl[prereset] instead.
Command Summary

\glsfirstplural\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle insert\rangle\} \quad \text{modifiers: } * + \langle alt-mod\rangle

glossaries

References the entry identified by \langle entry-label\rangle. The text produced is obtained from the firstplural value. The \langle insert\rangle argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. If you have defined the entry with \newabbreviation use \glsxtrfullpl for the full form or \glsxtrlongpl for the long form or use \glspl[prereset], as some abbreviation styles are too complicated to work with \glsfirstplural. For the first optional argument, see \glslink options.

\glsfirstpluralaccessdisplay\{\langle text\rangle\}\{\langle entry-label\rangle\}

glossaries-accsupp

Does \langle text\rangle with the firstpluralaccess replacement text (if set).

\glsfirstxpabbrvfont\{\langle text\rangle\}\{\langle category\rangle\}

glossaries-extra v1.49+

§4.5.3.1; 167

If the markshortwords attribute is set for the given category, this encapsulates \langle text\rangle with \glsfirstabbvfont otherwise with \glsfirstinnerfmtabbvfont. This command has to expand, so protect any content that shouldn’t expand.

\glsfirstxplongfont\{\langle category\rangle\}\{\langle text\rangle\}

glossaries-extra v1.49+

§4.5.3.1; 168

If the markwords attribute is set for the given category, this encapsulates \langle text\rangle with \glsfirstlongfont otherwise with \glsinnerfmtlongfont. This command has to expand, so protect any content that shouldn’t expand.

\GLSfmtfield\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}\{\langle internal-field\rangle\}

glossaries-extra v1.49+

§5.5.3; 246

As \glsfmtfield but changes the field value to all caps.

\Glsfmtfield\{\langle insert\rangle\}\{\langle cs\rangle\}\{\langle entry-label\rangle\}\{\langle internal-field\rangle\}

glossaries-extra v1.49+

§5.5.3; 246

As \glsfmtfield but uses \makefirstuc to change the field value to sentence case.

749
Command Summary

\texttt{\textbackslash glsfmtfield\{⟨insert⟩\}\{⟨cs⟩\}\{⟨entry-label⟩\}\{⟨internal-field⟩\}} glossaries-extra v1.49+

Applies the formatting command \texttt{⟨cs⟩} (which takes one argument) to the entry’s field value identified by the given internal field label, including \texttt{⟨insert⟩} appended. Used by the inner formatting commands. Note that \texttt{\textbackslash glsfmtfield} should not be robust as it needs to expand if it’s inside a case-changing command.

\texttt{\textbackslash GLSfmtfirst\{⟨entry-label⟩\}}

glossaries-extra v1.42+

For use within captions or section titles to display the formatted all caps \texttt{first}.

\texttt{\textbackslash Glsfmtfirst\{⟨entry-label⟩\}}

For use within captions or section titles to display the formatted sentence case \texttt{first}.

\texttt{\textbackslash glsfmtfirst\{⟨entry-label⟩\}}

For use within captions or section titles to display the formatted \texttt{first}.

\texttt{\textbackslash GLSfmtfirstpl\{⟨entry-label⟩\}}

glossaries-extra v1.42+

For use within captions or section titles to display the formatted all caps \texttt{firstplural}.

\texttt{\textbackslash Glsfmtfirstpl\{⟨entry-label⟩\}}

For use within captions or section titles to display the formatted sentence case \texttt{firstplural}.

\texttt{\textbackslash glsfmtfirstpl\{⟨entry-label⟩\}}

For use within captions or section titles to display the formatted \texttt{firstplural}. 750
Command Summary

\GLSfmtfull\{\textit{entry-label}\}\glossaries-extra v1.42+

Designed for use in section headings or captions, this expands to just \glspdffmtfull\{\textit{entry-label}\} in PDF bookmarks (no case-change), otherwise it expands to \GLSxtrtitle-full\{\textit{entry-label}\}.

\GLSfmtfullpl\{\textit{entry-label}\}\glossaries-extra v1.42+

Designed for use in section headings or captions, this expands to just \glspdffmtfullpl\{\textit{entry-label}\} in PDF bookmarks (no case-change), otherwise it expands to \GLSxtrtitle-fullpl\{\textit{entry-label}\}.

\Glsfmtfull\{\textit{entry-label}\}\glossaries-extra v1.02+

Designed for use in section headings or captions, this expands to just \glspdffmtfull\{\textit{entry-label}\} in PDF bookmarks (no case-change), otherwise it expands to \Glsxtrtitle\{\textit{entry-label}\}.

\Glsfmtfullpl\{\textit{entry-label}\}\glossaries-extra v1.02+

Designed for use in section headings or captions, this expands to just \glspdffmtfullpl\{\textit{entry-label}\} in PDF bookmarks (no case-change), otherwise it expands to \Glsxtrtitlefullpl\{\textit{entry-label}\}.
Designed for use in section headings or captions, this expands to just \glspdffmtfullpl\{⟨entry-label⟩\} in PDF bookmarks, otherwise it expands to \glsxtrtitlefullpl\{⟨entry-label⟩\}.

\GLSfmtinsert glossaries-extra v1.49+

As \glsfmtinsert but converts to all caps.

\glsfmtinsert glossaries-extra v1.49+

A shortcut that applies \glsxtrgenentrytextfmt to \glsinsert if \glsinsert isn’t empty.

\GLSfmtlong\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted all caps long form.

\Glsfmtlong\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted sentence case long form.

\glsfmtlong\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted long form.

\GLSfmtlongpl\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted all caps long plural form.

\Glsfmtlongpl\{⟨entry-label⟩\}
### Command Summary

For use within captions or section titles to display the formatted sentence case long plural form.

\glsfmtlongpl\{entry-label\}

For use within captions or section titles to display the formatted long plural form.

\GLSfmtname\{entry-label\}

For use within captions or section titles to display the formatted all caps name.

\Glsfmtname\{entry-label\}

For use within captions or section titles to display the formatted sentence case name.

\glsfmtname\{entry-label\}

For use within captions or section titles to display the formatted name.

\GLSfmtplural\{entry-label\}

For use within captions or section titles to display the formatted all caps plural.

\Glsfmtplural\{entry-label\}

For use within captions or section titles to display the formatted sentence case plural.

\glsfmtplural\{entry-label\}

For use within captions or section titles to display the formatted plural.
\GLSfmtshort\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted all caps short form.

\GLSfmtshortpl\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted all caps short plural form.

\Glsfmtshort\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted sentence case short form.

\Glsfmtshortpl\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted sentence case short plural form.

\glsfmtshort\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted short form.

\glsfmtshortpl\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted short plural form.

\GLSfmttext\{⟨entry-label⟩\}

For use within captions or section titles to display the formatted all caps text.
For use within captions or section titles to display the formatted sentence case text.

```
\glsfmttext{⟨entry-label⟩}
```

§5.3.2; 212

For use within captions or section titles to display the formatted text.

```
\glsforeachincategory[⟨glossary-types⟩]{⟨category⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

§10; 518

Iterates over all entry in the given list of glossaries (or all non-ignored glossaries, if the optional argument is omitted) and performs ⟨body⟩ for those entries that have the category set to ⟨category⟩. Within ⟨body⟩, the current entry can be referenced with ⟨label-cs⟩ and the glossary can be referenced with ⟨glossary-cs⟩.

```
\glsforeachwithattribute[⟨glossary-types⟩]{⟨attribute-label⟩}{⟨attribute-value⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}
```

§10; 518

Iterates over all entry in the given list of glossaries (or all non-ignored glossaries, if the optional argument is omitted) and performs ⟨body⟩ for those entries that have the attribute given by ⟨attribute-label⟩ set to ⟨attribute-value⟩. Within ⟨body⟩, the current entry can be referenced with ⟨label-cs⟩ and the glossary can be referenced with ⟨glossary-cs⟩.

```
\glsgenentryfmt glossaries v3.11a+
```

§5.5.5; 256

The display format used by \glsentryfmt for regular entries.

```
\glsgetattribute{⟨entry-label⟩}{⟨attribute⟩}
```

§10.2.2; 530

Expands to the value of the given attribute for the category associated with the entry identified by ⟨entry-label⟩. Expands to nothing if the attribute hasn’t been set.

```
\glsgetcategoryattribute{⟨category⟩}{⟨attribute⟩}
```

§10.2.2; 530
Command Summary

Expands to the value of the given attribute for the given category. Expands to nothing if the attribute hasn’t been set.

\texttt{\textbackslash glsgetwidestname} \hspace{1cm} \textit{glossaries-extra-stylemods v1.05+} \hfill§8.6.5.4; 440

Expands to the widest top-level name.

\texttt{\textbackslash glsgetwidestsubname\{\textlangle level\textrangle\}} \hspace{1cm} \textit{glossaries-extra-stylemods v1.05+} \hfill§8.6.5.4; 440

Expands to the widest name for the given hierarchical level or to the widest top-level name, if no widest name set for \textlangle level\textrangle.

\texttt{\textbackslash glsgroupheading\{\textlangle group-label\textrangle\}} \hspace{1cm} \textit{glossaries}

Inserted at the start of each group in a glossary (unless \texttt{groups=false}) to display the group’s heading, if applicable, using the title associated with \textlangle group-label\textrangle or, if no title provided, just \textlangle group-label\textrangle. This command is defined by glossary styles as appropriate.

\texttt{\textbackslash glsgroupskip} \hspace{1cm} \textit{glossaries}

Inserted before each group heading (except the first) in a glossary (unless \texttt{groups=false}). This command is defined by glossary styles as appropriate. Most of the predefined styles define this command to check the \texttt{nogroupskip} option.

\texttt{\textbackslash glshasattribute\{\textlangle entry-label\textrangle\}\{\textlangle attribute\textrangle\}\{\texttt{true}\}\{\texttt{false}\}} \hfill§10.2.2; 530

Tests if the given attribute has been set for the category associated with the entry identified by \textlangle entry-label\textrangle (using etoolbox’s \texttt{\textbackslash ifcsvoid}). Does \texttt{false} if the entry hasn’t been defined.

\texttt{\textbackslash glshascategoryattribute\{\textlangle category\textrangle\}\{\textlangle attribute\textrangle\}\{\texttt{true}\}\{\texttt{false}\}} \hfill§10.2.2; 530

Tests if the given attribute has been set for the given category (using etoolbox’s \texttt{\textbackslash ifcsvoid}).
\glshashchar \hspace{1em} \text{glossaries-extra-bib2gls v1.49+} \hspace{1em} §11.5.2; 560

Expands to a literal hash #.

\glshex\langle\text{hex}\rangle \hspace{1em} \text{glossaries-extra-bib2gls v1.21+} \hspace{1em} §11.5.2; 560

Expands to \string\u\langle\text{hex}\rangle.

\glshyperlink\langle\text{text}\rangle\{\langle\text{entry-label}\rangle\} \hspace{1em} \text{glossaries v1.17+}

Creates a hyperlink to the given entry with the hyperlink text provided in the optional argument. If omitted, the default is \glsetentrytext\langle\text{entry-label}\rangle.

\glshypernumber\langle\text{location(s)}\rangle \hspace{1em} \text{glossaries}

This will encapsulate each location with a hyperlink, if supported. This may be used as a location encap. The argument may be a single location or locations delimited by \delimR or \delimN. This command should not be used outside of location lists as it requires additional information in order to correctly form the hyperlinks.

\glsifapplyinnerfmtfield\langle\text{entry-label}\rangle\{\langle\text{internal-field}\rangle\}\{\langle\text{true}\rangle\}\{\langle\text{false}\rangle\} \hspace{1em} \text{glossaries-extra v1.49+} \hspace{1em} §5.5.3; 245

Tests if the field given by its internal field label \langle\text{internal-field}\rangle has been added to the inner formatting exclusion list for the entry identified by \langle\text{entry-label}\rangle.

\glsifattribute\langle\text{entry-label}\rangle\{\langle\text{attribute}\rangle\}\{\langle\text{value}\rangle\}\{\langle\text{true}\rangle\}\{\langle\text{false}\rangle\} \hspace{1em} §10.2.2; 530

Tests if the category associated with the entry identified by \langle\text{entry-label}\rangle has the given attribute set to \langle\text{value}\rangle. Does \langle\text{true}\rangle if the attribute is \langle\text{value}\rangle and \langle\text{false}\rangle otherwise. Does \langle\text{false}\rangle if there’s no such attribute for the given category or if the entry hasn’t been defined.
Tests if the category associated with the entry given by \textit{entry-label} has the given attribute set to \texttt{true}. Does \texttt{true} if the attribute is \texttt{true} and \texttt{false} otherwise. Does \texttt{false} if there’s no such attribute for the given category or if the entry hasn’t been defined.

Tests if the entry identified by \textit{entry-label} has the \texttt{category} set to \texttt{category} (uses \texttt{\textbackslash ifglsfieldeq} for the test).

Tests if the given category has the given attribute set to \texttt{value}. Does \texttt{true} if the attribute is \texttt{value} and \texttt{false} otherwise. Does \texttt{false} if there’s no such attribute for the given category.

Does \texttt{true} if the category has the attribute (whose value is a comma-separated list) contains the given item and \texttt{false} otherwise. Does \texttt{false} if there’s no such attribute for the given category. The item and list are expanded and passed to datatool’s \texttt{\DTLifinlist} to perform the test.

Tests if the given category has the given attribute set to \texttt{true}. Does \texttt{true} if the attribute is \texttt{true} and \texttt{false} otherwise. Does \texttt{false} if there’s no such attribute for the given category.

Tests if the value obtained from \texttt{\glsentryindexcount} is greater than 0.
\textbf{Command Summary}

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsifnotregular{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}</td>
<td>Does \texttt{(true)} if the category for the given entry has the \texttt{regular} attribute explicitly set to \texttt{false}, otherwise does \texttt{(false)}.</td>
</tr>
<tr>
<td>\glsifnotregularcategory{⟨category⟩}{⟨true⟩}{⟨false⟩}</td>
<td>Does \texttt{(true)} if the given category has the \texttt{regular} attribute explicitly set to \texttt{false}, otherwise does \texttt{(false)}.</td>
</tr>
<tr>
<td>\glsifplural{⟨true⟩}{⟨false⟩}</td>
<td>Initialised by the \texttt{\glstex}-like and \texttt{\glstext}-like commands, this expands to \texttt{(true)} if the calling command accesses a plural field (such as \texttt{\glsp1} or \texttt{\glspplural}) otherwise it expands to \texttt{(false)}. This command may be used within associated hooks, entry display styles (\texttt{\defglsentryfmt}), and the post-link hook.</td>
</tr>
<tr>
<td>\glsifregular{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}</td>
<td>Does \texttt{(true)} if the category for the given entry has the \texttt{regular} attribute explicitly set to \texttt{true}, otherwise does \texttt{(false)}.</td>
</tr>
<tr>
<td>\glsifregularcategory{⟨category⟩}{⟨true⟩}{⟨false⟩}</td>
<td>Does \texttt{(true)} if the given category has the \texttt{regular} attribute explicitly set to \texttt{true}, otherwise does \texttt{(false)}.</td>
</tr>
<tr>
<td>\glsignore{⟨text⟩}</td>
<td>Does nothing. When used as a location encap, this signifies to \texttt{bib2gls} that the entry is required but the location shouldn’t be added to the location list. With other indexing methods, this simply creates an invisible location.</td>
</tr>
</tbody>
</table>
**Command Summary**

\texttt{\textbackslash glsindexingsetting} \hspace{1cm} glossaries v4.50+ & glossaries-extra v1.49+

Indicates what indexing option has been chosen.

\texttt{\textbackslash glsindexsubgroupitem}\{⟨previous group level⟩\}⟨level⟩\{⟨parent label⟩\}⟨group label⟩\{⟨group title⟩\} \hspace{1cm} glossaries-extra-stylemods v1.49+

Used to format sub-group headers for the indexgroup styles.

\texttt{\textbackslash glsinitreunsets} \hspace{1cm} glossaries-extra v1.49+

Hook that initialises the \texttt{prereset}, \texttt{preunset} and \texttt{postunset} settings.

\texttt{\textbackslash glsinlinedescformat}\{⟨description⟩\}⟨symbol⟩\{⟨location list⟩\} \hspace{1cm} glossary-inline v3.03+

Formats the description, symbol and location list for top-level entries.

\texttt{\textbackslash glsinlinesubdescformat}\{⟨description⟩\}⟨symbol⟩\{⟨location list⟩\} \hspace{1cm} glossary-inline v3.03+

Formats the description, symbol and location list for child entries.

\texttt{\textbackslash glsinnerfmtabbrvfont}\{⟨text⟩\} \hspace{1cm} glossaries-extra v1.49+

Robust command that applies both \texttt{\textbackslash glsabbrvfont} and \texttt{\textbackslash glsxtrgenentrytextfmt} to \texttt{⟨text⟩}.

\texttt{\textbackslash glsinnerfmtlongfont}\{⟨text⟩\} \hspace{1cm} glossaries-extra v1.49+

Applies both \texttt{\textbackslash glslongfont} and \texttt{\textbackslash glsxtrgenentrytextfmt} to \texttt{⟨text⟩}.
## Command Summary

### \glsinsert

The final \textit{⟨insert⟩} argument passed to the \gls-like commands (but not to the \glstext-like commands, where the \textit{⟨insert⟩} is added to \glscustomtext). This command may be used within associated hooks, entry display styles (\texttt{\defglsentryfmt}), and the post-link hook.

### \glstok

A token register that stores the options passed to \texttt{\newabbreviation}.

### \glslabel

The current entry label, initialised by the \gls-like and \glstext-like commands. This command may be used within associated hooks, entry display styles (\texttt{\defglsentryfmt}), and the post-link hook.

### \glsletentryfield\{⟨cs⟩\}{⟨entry-label⟩}{⟨field-label⟩}

Fetches the value of the given field (identified by its internal label \textit{⟨field-label⟩}) for the entry given by \textit{⟨entry-label⟩} and stores it in the command \textit{⟨cs⟩}.

### \Glslink\[\langle options\rangle\]{⟨entry-label⟩}{⟨text⟩}

As \texttt{\glslink} but sets the link text to \texttt{\glssentencecase\{⟨text⟩\}}. This is provided to allow a sentence case mapping in the event that \texttt{\glslink} occurs at the start of content that has automated case-changing.
### Command Summary

References the entry identified by \textit{⟨entry-label⟩} with the given \textit{⟨text⟩} as the link text. This command does not alter or depend on the first use flag (use \texttt{\glsdisp} instead, if the first use flag needs to be unset). This command is considered a \texttt{\glstext}-like command. For the first optional argument, see \texttt{\glslink} options.

\begin{itemize}
  \item \texttt{\glslinkcheckfirsthyperhook} \quad \textit{glossaries v4.08+}
  \begin{footnotesize}
    Hook used at the end of the code in the \texttt{\gls}-like commands that tests if the hyperlink should be switched off on first use.
  \end{footnotesize}
  \item \texttt{\glslinkpostsetkeys} \quad \textit{glossaries v4.16+}
  \begin{footnotesize}
    Hook implemented after setting the options passed to the \texttt{\gls}-like and \texttt{\glstext}-like commands.
  \end{footnotesize}
  \item \texttt{\glslinkpresetkeys} \quad \textit{glossaries-extra v1.26+}
  \begin{footnotesize}
    Hook implemented before setting the options passed to the \texttt{\gls}-like and \texttt{\glstext}-like commands.
  \end{footnotesize}
  \item \texttt{\glslinkwrcontent\{⟨code⟩\}} \quad \textit{glossaries-extra v1.48+}
  \begin{footnotesize}
    Encapsulates the link text and indexing. Just does \textit{⟨code⟩} by default.
  \end{footnotesize}
  \item \texttt{\glslistchildpostlocation} \quad \textit{initial: .} \quad \textit{glossaries-extra-stylemods v1.21+}
  \begin{footnotesize}
    Used after the child entry location list for the list styles.
  \end{footnotesize}
  \item \texttt{\glslistchildprelocation} \quad \textit{initial: \glslistprelocation}
  \begin{footnotesize}
    \textit{glossaries-extra-stylemods v1.21+}
  \end{footnotesize}
  \begin{footnotesize}
    Used before the child entry location list for the list styles.
  \end{footnotesize}
\end{itemize}

\section*{762}
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glslistdesc{⟨entry-label⟩}</td>
<td>Used to display the description for the list styles.</td>
</tr>
<tr>
<td>\glslistdottedwidth</td>
<td>A length register used by listdotted.</td>
</tr>
<tr>
<td>\glslistexpandedname{⟨entry-label⟩}</td>
<td>Used by \glslistinit to provide better integration with gettitlestring.</td>
</tr>
<tr>
<td>\glslistgroupafterheader</td>
<td>Used after group headings in the listgroup styles.</td>
</tr>
<tr>
<td>\glslistgroupheaderitem{⟨group-label⟩}{⟨header code⟩}</td>
<td>Used to display the group headings in the listgroup styles.</td>
</tr>
<tr>
<td>\glslistgroupskip</td>
<td>Used for the group skip in the list styles.</td>
</tr>
<tr>
<td>\glslistinit</td>
<td>Used to disable problematic commands at the start the list styles to provide better integration with gettitlestring.</td>
</tr>
</tbody>
</table>
Command Summary

Used to display the top-level entry item in the list styles.

\texttt{\textbackslash glslistprelocation} \emph{initial: \textbackslash glsxtrprelocation} \texttt{glossaries-extra-stylemods v1.21+}

§8.6.5.3; 434

Used before the top-level entry location list for the list styles.

\texttt{\textbackslash glslocalreset\{}\langle entry-label\rangle\}\texttt{glossaries}

Locally resets the entry’s first use flag. That is, this marks the entry as “not used”.

\texttt{\textbackslash glslocalresetall[\langle glossary labels list\rangle]} \texttt{glossaries}

Locally resets the first use flag for all entries in whose labels are listed in the \langle glossary labels list\rangle comma-separated list. If the optional argument is omitted, the list of all non-ignored glossaries is assumed.

\texttt{\textbackslash glslocalreseteach\{}\langle entry-labels\rangle\}\texttt{glossaries-extra v1.31+}

§5.10; 284

Locally resets each listed entry’s first use flag.

\texttt{\textbackslash glslocalunset\{}\langle entry-label\rangle\}\texttt{glossaries}

Locally unset\texttt{s the entry’s first use flag. That is, this marks the entry as “used”}.

\texttt{\textbackslash glslocalunsetall[\langle glossary labels list\rangle]} \texttt{glossaries}

Locally unset\texttt{s the first use flag for all entries in whose labels are listed in the \langle glossary labels list\rangle comma-separated list. If the optional argument is omitted, the list of all non-ignored glossaries is assumed.
Locally unsets each listed entry’s first use flag.

Does \textit{<text>} with the \texttt{longaccess} replacement text (if set).

Formatting command for the long form used by the abbreviation styles that don’t apply a font change by default.

Long form font used by the “em” abbreviation styles.

Expands to the column alignment for the first custom field.

Expands to the internal field name of the first custom field.

The format of the first custom entry.
Command Summary

Expands to the header name of the first custom column.

\glslongextraCustomIIAlign
\textit{initial:}\ l\ glossary\-longextra v1.50+

Expands to the column alignment for the second custom field.

\glslongextraCustomIIField
\textit{initial: userii}\ glossary\-longextra v1.50+

Expands to the internal field name of the second custom field.

\glslongextraCustomIIFmt{\langle entry-label\rangle}\ glossary\-longextra v1.50+

The format of the second custom entry.

Expands to the header name of the second custom column.

\glslongextraCustomIIIFormat
\textit{initial:}\ l\ glossary\-longextra v1.50+

Expands to the column alignment for the third custom field.

\glslongextraCustomIIIField
\textit{initial: useriii}\ glossary\-longextra v1.50+

Expands to the internal field name of the third custom field.

\glslongextraCustomIIIFmt{\langle entry-label\rangle}\ glossary\-longextra v1.50+

The format of the third custom entry.
Expands to the header name of the third custom column.

\texttt{\textbackslash glslongextraCustomIIIHeader} \hspace{2cm} \textit{glossary-longextra v1.50+} §8.7.2.7; 468

The header for the \texttt{longtable long-custom3-name} style.

\texttt{\textbackslash glslongextraCustomIIINameHeader} \hspace{2cm} \textit{glossary-longextra v1.50+} §8.7.2.7; 470

\texttt{\textbackslash glslongextraCustomIIINameTabularHeader} \hspace{2cm} \textit{glossary-longextra v1.50+} §8.7.2.7; 471

Used to set the length \texttt{\glsdescwidth} for \texttt{long-name-custom3-desc} style.

\texttt{\textbackslash glslongextraCustomIIISetDescWidth} \hspace{2cm} \textit{glossary-longextra v1.50+} §8.7.2.7; 472

The header for the \texttt{longtable long-custom2-name} style.

\texttt{\textbackslash glslongextraCustomINameHeader} \hspace{2cm} \textit{glossary-longextra v1.50+} §8.7.2.7; 473

\texttt{\textbackslash glslongextraCustomINameTabularHeader} \hspace{2cm} \textit{glossary-longextra v1.50+} §8.7.2.7; 474

The header for the \texttt{longtable long-custom2-name} style.

\texttt{\textbackslash glslongextraCustomIIISetDescWidth} \hspace{2cm} \textit{glossary-longextra v1.50+} §8.7.2.7; 475

Used to set the length \texttt{\glsdescwidth} for \texttt{long-name-custom2-desc} style.

\texttt{\textbackslash glslongextraCustomINameHeader} \hspace{2cm} \textit{glossary-longextra v1.50+} §8.7.2.7; 476

The header for the \texttt{longtable long-custom1-name} style.

767
Command Summary

\glslongextraCustomINameTabularHeader  
\textit{glossary-longextra v1.50+}
§8.7.2.7; 470

The header for the long–custom1–name style.

\glslongextraCustomISetDescWidth  
\textit{glossary-longextra v1.50+}
§8.7.2.7; 472

Used to set the length \glsdescwidth for long–name–custom1–desc style.

\glslongextraCustomTabularFooter  
\textit{glossary-longextra v1.50+}
§8.7.2.7; 469

The footer for the custom styles.

\glslongextraDescAlign  
\textit{glossary-longextra v1.37+}
§8.7.2; 452

The horizontal alignment for the description column.

\glslongextraDescCustomIIINameHeader  
\textit{glossary-longextra v1.50+}
§8.7.2.7; 474

The header for the longtable long–desc–custom3–name style.

\glslongextraDescCustomIIINameTabularHeader  
\textit{glossary-longextra v1.50+}
§8.7.2.7; 474

The header for the longtable long–desc–custom3–name style.

\glslongextraDescCustomIINameHeader  
\textit{glossary-longextra v1.50+}
§8.7.2.7; 474

The header for the longtable long–desc–custom2–name style.

\glslongextraDescCustomIINameTabularHeader  
\textit{glossary-longextra v1.50+}
§8.7.2.7; 474

The header for the longtable long–desc–custom2–name style.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\textbackslash glacslongextraDescCustomINameHeader}</td>
<td>The header for the longtable long-desc-custom1-name style.</td>
</tr>
<tr>
<td>\texttt{\textbackslash glacslongextraDescCustomINameTabularHeader}</td>
<td>The header for the long-desc-custom1-name style.</td>
</tr>
<tr>
<td>\texttt{\textbackslash glacslongextraDescFmt{⟨entry-label⟩}}</td>
<td>Used by the glossary-longextra styles to display a top-level entry’s description and post-description hook.</td>
</tr>
<tr>
<td>\texttt{\textbackslash glacslongextraDescNameHeader}</td>
<td>Sets the header and footer for the long-desc-name style with longtable.</td>
</tr>
<tr>
<td>\texttt{\textbackslash glacslongextraDescNameTabularFooter}</td>
<td>Displays the footer for the long-desc-name style.</td>
</tr>
<tr>
<td>\texttt{\textbackslash glacslongextraDescNameTabularHeader}</td>
<td>Displays the header for the long-desc-name style.</td>
</tr>
<tr>
<td>\texttt{\textbackslash glacslongextraDescSymHeader}</td>
<td>Sets the header and footer for the long-desc-sym style with longtable.</td>
</tr>
</tbody>
</table>

---

769
Sets the header and footer for the long-desc-sym-name style with longtable.

\glslongextraDescSymNameTabularFooter\par §8.7.2.2; 458

Displays the footer for the long-desc-sym-name style.

\glslongextraDescSymNameTabularHeader\par §8.7.2.2; 457

Displays the header for the long-desc-sym-name style.

\glslongextraDescSymTabularFooter\par §8.7.2.5; 464

Displays the footer for the long-desc-sym style.

\glslongextraDescSymTabularHeader\par §8.7.2.5; 463

Displays the header for the long-desc-sym style.

\glslongextraGroupHeading\par §8.7.2; 454

Formats the top-level group heading.

\glslongextraHeaderFmt\par §8.7.2; 451

Used to format the column headers.

\glslongextraLocationAlign\par §8.7.2; 453

The horizontal alignment for the location list column.
Sets the header and footer for the long-loc-desc-name style with longtable.

Displays the footer for the long-loc-desc-name style.

Displays the header for the long-loc-desc-name style.

Sets the header and footer for the long-loc-desc-sym-name style with longtable.

Displays the footer for the long-loc-desc-sym-name style.

Displays the header for the long-loc-desc-sym-name style.

Used by the glossary-longextra styles to display a top-level entry’s location list.

Sets the header and footer for the long-loc-sym-desc-name style with longtable.
## Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glslongextraLocationSymDescNameTabularFooter</td>
<td>Displays the footer for the long-loc-sym-desc-name style.</td>
<td>§8.7.2.4; 461</td>
</tr>
<tr>
<td>\glslongextraLocationSymDescNameTabularHeader</td>
<td>Displays the header for the long-loc-sym-desc-name style.</td>
<td>§8.7.2.4; 461</td>
</tr>
<tr>
<td>\glslongextraLocSetDescWidth</td>
<td>Computes the value of \glsdescwidth according to the widest name for styles that only show the name, location list and description.</td>
<td>§8.7.2.3; 458</td>
</tr>
<tr>
<td>\glslongextraLongFmt{⟨entry-label⟩}</td>
<td>The formatting for the long form in the abbr-long-short and abbr-short-long styles.</td>
<td>§8.7.2.6; 465</td>
</tr>
<tr>
<td>\glslongextraLongHeader initial: \descriptionname</td>
<td>The long column header for the abbr-long-short and abbr-short-long styles.</td>
<td>§8.7.2.6; 464</td>
</tr>
<tr>
<td>\glslongextraLongShortHeader</td>
<td>Sets the header and footer for the abbr-short-long style with longtable.</td>
<td>§8.7.2.6; 467</td>
</tr>
<tr>
<td>\glslongextraLongShortTabularFooter</td>
<td>Displays the footer for the abbr-short-long style.</td>
<td>§8.7.2.6; 466</td>
</tr>
<tr>
<td>\glslongextraLongShortTabularHeader</td>
<td></td>
<td>§8.7.2.6; 466</td>
</tr>
</tbody>
</table>

772
Displays the header for the abbr-short-long style.

\texttt{\textbackslash glslongextraNameAlign} \hspace{1cm} \textit{initial: 1} glossary-longextra v1.37+

The horizontal alignment for the name column.

\texttt{\textbackslash glslongextraNameCustomIDescHeader} glossary-longextra v1.50+

The header for the longtable long-name-custom1-desc style.

\texttt{\textbackslash glslongextraNameCustomIDescTabularHeader} glossary-longextra v1.50+

The header for the long-name-custom1-desc style.

\texttt{\textbackslash glslongextraNameCustomIHeader} glossary-longextra v1.50+

The header for the longtable long-name-custom1 style.

\texttt{\textbackslash glslongextraNameCustomIIDescHeader} glossary-longextra v1.50+

The header for the longtable long-name-custom2-desc style.

\texttt{\textbackslash glslongextraNameCustomIIDescTabularHeader} glossary-longextra v1.50+

The header for the long-name-custom2-desc style.

\texttt{\textbackslash glslongextraNameCustomIIHeader} glossary-longextra v1.50+

The header for the longtable long-name-custom2 style.
The header for the longtable long-name-custom3-desc style.

The header for the long-name-custom3-desc style.

The header for the longtable long-name-custom3 style.

The header for the long-name-custom3 style.

The header for the long-name-custom2 style.

The header for the long-name-custom1 style.

Sets the header and footer for the long-name-desc style with longtable.

Sets the header and footer for the long-name-desc-loc style with longtable.
Displays the footer for the long-name-desc-loc style.

Displays the header for the long-name-desc-loc style.

Sets the header and footer for the long-name-desc-sym style with longtable.

Sets the header and footer for the long-name-desc-sym-loc style with longtable.

Displays the footer for the long-name-desc-sym-loc style.

Displays the header for the long-name-desc-sym-loc style.

Displays the footer for the long-name-desc-sym style.

Displays the header for the long-name-desc-sym style.
## Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glslongextraNameDescTabularFooter</td>
<td>Displays the footer for the long-name-desc style.</td>
</tr>
<tr>
<td>\glslongextraNameDescTabularHeader</td>
<td>Displays the header for the long-name-desc style.</td>
</tr>
<tr>
<td>\glslongextraNameFmt{⟨entry-label⟩}</td>
<td>Used by the glossary-longextra styles to add the hypertarget (if supported) and display a top-level entry’s name.</td>
</tr>
<tr>
<td>\glslongextraNameSymDescHeader</td>
<td>Sets the header and footer for the long-name-sym-desc style with longtable.</td>
</tr>
<tr>
<td>\glslongextraNameSymDescLocationHeader</td>
<td>Sets the header and footer for the long-name-sym-desc-loc style with longtable.</td>
</tr>
<tr>
<td>\glslongextraNameSymDescLocationTabularFooter</td>
<td>Displays the footer for the long-name-sym-desc-loc style.</td>
</tr>
<tr>
<td>\glslongextraNameSymDescLocationTabularHeader</td>
<td>Displays the header for the long-name-sym-desc-loc style.</td>
</tr>
</tbody>
</table>

776
Command Summary

Displays the footer for the long-name-sym-desc style.

\glslongextraNameSymDescTabularFooter\glslongextraNameSymDescTabularFooter glossary-longextra v1.37+

Displays the header for the long-name-sym-desc style.

\glslongextraSetDescWidth\glslongextraSetDescWidth glossary-longextra v1.37+

Computes the value of \glsdescwidth according to the widest name for styles that only show the name and description.

\glslongextraSetWidest{\langle widest-name\rangle}\glslongextraSetWidest{\langle widest-name\rangle} glossary-longextra v1.37+

Identifies \langle widest-name\rangle as the widest top-level name.

\glslongextraShortHeader \glslongextraShortHeader initial: \entryname glossary-longextra v1.49+

The short column header for the abbr-long-short and abbr-short-long styles.

\glslongextraShortLongHeader\glslongextraShortLongHeader glossary-longextra v1.49+

Sets the header and footer for the abbr-short-long style with longtable.

\glslongextraShortLongTabularFooter\glslongextraShortLongTabularFooter glossary-longextra v1.49+

Displays the footer for the abbr-short-long style.

\glslongextraShortLongTabularHeader\glslongextraShortLongTabularHeader glossary-longextra v1.49+

Displays the header for the abbr-short-long style.
Sets the value of `\glsdescwidth` for the abbr–long–short and abbr–short–long styles.

The formatting, including the target, for the short form in the abbr–long–short and abbr–short–long styles.

The format of the first custom sub-entry.

The format of the second custom sub-entry.

The format of the third custom sub-entry.

Used by the `glossary-longextra` styles to display a child entry’s description and post-description hook.

Formats the sub-group heading, if supported.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Summary</th>
<th>Version</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\glslongextraSubLocationFmt{⟨level⟩}{⟨entry-label⟩}{⟨location list⟩}</code></td>
<td>Used by the glossary-longextra styles to display a child entry’s location list.</td>
<td>§8.7.2; 453</td>
<td></td>
</tr>
<tr>
<td><code>\glslongextraSubLongFmt{⟨level⟩}{⟨entry-label⟩}</code></td>
<td>The formatting for child entry long forms in the abbr-long-short and abbr-short-long styles.</td>
<td>§8.7.2.6; 465</td>
<td></td>
</tr>
<tr>
<td><code>\glslongextraSubNameFmt{⟨level⟩}{⟨entry-label⟩}</code></td>
<td>Used by the glossary-longextra styles to add the hypertarget (if supported) for child-entries. The name isn’t shown by default.</td>
<td>§8.7.2; 452</td>
<td></td>
</tr>
<tr>
<td><code>\glslongextraSubShortTargetFmt{⟨level⟩}{⟨entry-label⟩}</code></td>
<td>The formatting, including the target, for child entry short forms in the abbr-long-short and abbr-short-long styles.</td>
<td>§8.7.2.6; 465</td>
<td></td>
</tr>
<tr>
<td><code>\glslongextraSubSymbolFmt{⟨level⟩}{⟨entry-label⟩}</code></td>
<td>Used by the glossary-longextra styles to display a child entry’s symbol.</td>
<td>§8.7.2; 453</td>
<td></td>
</tr>
<tr>
<td><code>\glslongextraSubSymbolOrName{⟨level⟩}{⟨entry-label⟩}</code></td>
<td>Adds the hypertarget (if supported) and displays the symbol if set or the name otherwise for child entries.</td>
<td>§8.7.2.5; 462</td>
<td></td>
</tr>
</tbody>
</table>

779
Command Summary

Adds the hypertarget (if supported) and displays the symbol for child entries.

\texttt{\textbackslash glslongextraSymbolAlign} \textit{initial}: \texttt{c} \quad \text{glossary–longextra v1.37+} \hspace{1cm} §8.7.2; 454

The horizontal alignment for the symbol column.

\texttt{\textbackslash glslongextraSymbolFmt\{\textit{entry-label}\}} \quad \text{glossary–longextra v1.37+} \hspace{1cm} §8.7.2; 453

Used by the \texttt{glossary–longextra} styles to display a top-level entry’s symbol.

\texttt{\textbackslash glslongextraSymbolNameAlign} \textit{initial}: \texttt{l} \quad \text{glossary–longextra v1.49+} \hspace{1cm} §8.7.2.5; 462

The horizontal alignment for the symbol column when it’s being used instead of the name.

\texttt{\textbackslash glslongextraSymbolOrName\{\textit{entry-label}\}} \quad \text{glossary–longextra v1.49+} \hspace{1cm} §8.7.2.5; 462

Adds the hypertarget (if supported) and displays the symbol if set or the name otherwise for top-level entries.

\texttt{\textbackslash glslongextraSymbolTargetFmt\{\textit{entry-label}\}} \quad \text{glossary–longextra v1.49+} \hspace{1cm} §8.7.2.5; 462

Adds the hypertarget (if supported) and displays the symbol for top-level entries.

\texttt{\textbackslash glslongextraSymDescHeader} \quad \text{glossary–longextra v1.49+} \hspace{1cm} §8.7.2.5; 463

Sets the header and footer for the \texttt{long–sym–desc} style with \texttt{longtable}.

\texttt{\textbackslash glslongextraSymDescNameHeader} \quad \text{glossary–longextra v1.37+} \hspace{1cm} §8.7.2.2; 457

Sets the header and footer for the \texttt{long–sym–desc–name} style with \texttt{longtable}.

780
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\glslongextraSymDescNameTabularFooter}</td>
<td>Displays the footer for the long-sym-desc-name style.</td>
</tr>
<tr>
<td>\texttt{\glslongextraSymDescNameTabularHeader}</td>
<td>Displays the header for the long-sym-desc-name style.</td>
</tr>
<tr>
<td>\texttt{\glslongextraSymDescTabularFooter}</td>
<td>Displays the footer for the long-sym-desc style.</td>
</tr>
<tr>
<td>\texttt{\glslongextraSymDescTabularHeader}</td>
<td>Displays the header for the long-sym-desc style.</td>
</tr>
<tr>
<td>\texttt{\glslongextraSymLocSetDescWidth}</td>
<td>Computes the value of \texttt{\glsdescwidth} according to the widest name for styles that show the name, symbol, location list and description.</td>
</tr>
<tr>
<td>\texttt{\glslongextraSymNoNameSetDescWidth}</td>
<td>Computes the value of \texttt{\glsdescwidth} according to the widest name for styles that only show the symbol and description.</td>
</tr>
<tr>
<td>\texttt{\glslongextraSymSetDescWidth}</td>
<td>Computes the value of \texttt{\glsdescwidth} according to the widest name for styles that only show the name, symbol and description.</td>
</tr>
</tbody>
</table>
Command Summary

\texttt{\textbackslash glslongextraTabularVAlign} \hspace{1cm} initial: \texttt{c} \hspace{1cm} \texttt{glossary-longextra v1.37+} \hspace{1cm} §8.7.2; 451

Only for use with the \texttt{tabular} setting, this should expand to the \texttt{tabular} environment’s vertical alignment specifier.

\texttt{\textbackslash glslongextraUpdateWidest\{\langle name\rangle\}} \hspace{1cm} \texttt{glossary-longextra v1.37+} \hspace{1cm} §8.7.2; 453

If \langle name\rangle is wider than the current widest name, it will be set as the new widest name.

\texttt{\textbackslash glslongextraUpdateWidestChild\{\langle level\rangle\}\{\langle name\rangle\}} \hspace{1cm} \texttt{glossary-longextra v1.37+} \hspace{1cm} §8.7.2; 453

As \texttt{glslongextraUpdateWidest} but for child entries. Does nothing by default.

\texttt{\textbackslash GlsLongExtraUseTabularfalse} \hspace{1cm} \texttt{glossary-longextra v1.37+} \hspace{1cm} §8.7.2; 451

Sets \texttt{\ifGlsLongExtraUseTabular} to false (if this setting is required, the style must be set after this command).

\texttt{\textbackslash GlsLongExtraUseTabulartrue} \hspace{1cm} \texttt{glossary-longextra v1.37+} \hspace{1cm} §8.7.2; 450

Sets \texttt{\ifGlsLongExtraUseTabular} to true (if this setting is required, the style must be set after this command).

\texttt{\textbackslash glslongfont\{\langle text\rangle\}} \hspace{1cm} 172

Font formatting command for the long form, initialised by the abbreviation style.

\texttt{\textbackslash glslongfootnotefont\{\langle text\rangle\}} \hspace{1cm} \texttt{glossaries-extra v1.05+} \hspace{1cm} 145

Formatting command for the long form used by the footnote abbreviation styles.
Command Summary

\texttt{\textbackslash glslonghyphenfont\{\textit{text}\}} \hspace{1cm} glossaries-extra v1.17+

Long form font used by the “hyphen” abbreviation styles.

\texttt{\textbackslash glslongonlyfont\{\textit{text}\}} \hspace{1cm} glossaries-extra v1.17+

Long form font used by the “only” abbreviation styles.

\texttt{\textbackslash glslongpltok}

A token register that stores the long plural form (which may have been modified after being passed to \texttt{\textbackslash newabbreviation}).

\texttt{\textbackslash glslongpluralaccessdisplay\{\textit{text}\}\{\textit{entry-label}\}} \hspace{1cm} glossaries-accsupp

Does \texttt{(text)} with the \texttt{longpluralaccess} replacement text (if set).

\texttt{\textbackslash glslongtok}

A token register that stores the long form (which may have been modified after being passed to \texttt{\textbackslash newabbreviation}).

\texttt{\textbackslash glslonguserfont\{\textit{text}\}} \hspace{1cm} glossaries-extra v1.04+

Long form font used by the “user” abbreviation styles.

\texttt{\textbackslash glslowercase\{\textit{text}\}} \hspace{1cm} glossaries v4.50+

Converts \texttt{(text)} to lowercase.

\texttt{\textbackslash glsmakefirstuc\{\textit{text}\}} \hspace{1cm} mfirstuc v1.05+

§4.5.3.1; 164

§5.2.2; 200

783
Command Summary

Used by \makefirstuc to perform the actual case-change. As from mfirstuc v2.08+ this just uses \MFUsentencecase.

\glsmfuaddmap\{⟨cs1⟩\}{⟨cs2⟩} \text{ glossaries v4.50+ & glossaries-extra v1.49+}

If mfirstuc v2.08+ is installed, this will use \MFUaddmap, otherwise it will use \glsmfuexcl instead. See §5.2.1 for further details.

\glsmfublocker\{⟨cs⟩\} \text{ glossaries v4.50+ & glossaries-extra v1.49+}

If mfirstuc v2.08+ is installed, this will use \MFUblocker, otherwise it will use \glsmfuexcl instead. See §5.2.1 for further details.

\glsmfuexcl\{⟨cs⟩\} \text{ glossaries v4.50+ & glossaries-extra v1.49+}

If mfirstuc v2.08+ is installed, this will use \MFUexcl, otherwise it will implement something similar. See §5.2.1 for further details.

\GLSname[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} \text{ modifiers: * + ⟨alt-mod⟩ glossaries}

As \glsname but converts the link text to all caps. This command is incompatible with some abbreviation styles.

\Glsname[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} \text{ modifiers: * + ⟨alt-mod⟩ glossaries}

As \glsname but converts the link text to sentence case. Use \Glossentryname within custom glossary styles instead of this command.

\glsname[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} \text{ modifiers: * + ⟨alt-mod⟩ glossaries}

References the entry identified by ⟨entry-label⟩. The text produced is obtained from the name value. The ⟨insert⟩ argument will be inserted at the end of the link text. This command
Command Summary

does not alter or depend on the first use flag. For the first optional argument, see \glslink options. Use \glossentryname within custom glossary styles instead of this command.

\glsnameaccessdisplay\{⟨text⟩\}\{⟨entry-label⟩\}  glossaries–accsupp

Does ⟨text⟩ with the access replacement text (if set).

\glsnamefont\{⟨text⟩\}  glossaries

Used by \glossentryname to apply a font change to the name, unless (with glossaries–extra) the \glossnamefont attribute has been set.

\glsnextpages  glossaries v3.0+

Designed for use with makeindex and xindy, this may be placed in an entry’s description to override nonumberlist.

\glsnoexpandfields  glossaries v3.08a+

Don’t expand field values when defining entries, except for those that explicitly have expansion enabled with \glssetexpandfield.

\glsnoidxdisplayloc\{⟨prefix⟩\}\{⟨counter⟩\}\{⟨format⟩\}\{⟨location⟩\}  glossaries v4.04+

§11.5.6; 582

Used to display a location in the location list.

\glsnonextpages  glossaries v1.17+

Designed for use with makeindex and xindy, this may be placed in an entry’s description to suppress the entry’s location list.

\glsnumberformat\{⟨location(s)⟩\}  glossaries
The default format for entry locations. If hyperlinks are defined, this will use \glshypernumber otherwise it will simply display its argument, which may be a single location, or locations delimited by \delimR or \delimN.

\glsnumbersgroupname\initial: Numbers glossaries

Expands to the title of the numbers group and (if the numbers package option is used) the numbers glossary.

\glspagelistwidth
glossary-long & glossary-super

A length register used to set the width of the location list column for tabular-like styles.

\glspar
glossaries

Paragraph break (for instances where \par can’t be used directly).

\glspatchLToutput
glossary-longbooktabs v4.21+

Applies a patch to longtable to check for instances of the group skip occurring at a page break.

\glspdffmtfull\{\entry-label\}\glossaries-extra v1.42+\§5.3.2; 210

Shortcut for \glsentrylong\{\entry-label\} \glsentryshort\{\entry-label\} for use in PDF bookmarks or other text-only contexts.

\glspdffmtfullpl\{\entry-label\}\glossaries-extra v1.42+\§5.3.2; 210

Shortcut for \glsentrylongpl\{\entry-label\} \glsentryshortpl\{\entry-label\} for use in PDF bookmarks or other text-only contexts.
Command Summary

\glspenaltygroupskip \hspace{1cm} glossary-longbooktabs v4.21+

The definition of \gls{glsgroupskip} with \texttt{nogroupskip=false} for the glossary-longbooktabs styles.

\glspercentchar \hspace{1cm} glossaries v4.10+

Expands to a literal percent sign.

\GLSp{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩ glossaries

As \glspl{glspl} but converts the link text to all caps.

\Glsp{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩ glossaries

As \glspl{glspl} but converts the first character of the link text to uppercase (for the start of a sentence) using \texttt{\makefirstuc}.

\glspl{glspl}⟩{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩ glossaries

As \gls{gls} but uses the relevant plural form.

\GLSplural{⟨options⟩}{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩ glossaries

As \glsplural{glsplural} but converts the link text to all caps. If you have defined the entry with \texttt{\newabbreviation} use \texttt{\GLSxtrshortpl} or \texttt{\GLSp[preunset]} instead.
**Command Summary**

As `\glsplural` but converts the first character of the link text to uppercase (for the start of a sentence) using `\makefirstuc`. If you have defined the entry with `\newabbreviation` use `\Glsxtrshortpl` or `\Glspl[preunset]` instead.

`\glsplural[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} modifiers: * + ⟨alt-mod⟩`

References the entry identified by `⟨entry-label⟩`. The text produced is obtained from the plural value. The `⟨insert⟩` argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. If you have defined the entry with `\newabbreviation` use `\glsxtrshortpl` for the short form or `\gls[preunset]`, as some abbreviation styles are too complicated to work with `\glsplural`. For the first optional argument, see `\glslink` options.

`\glspluralaccessdisplay{⟨text⟩}{⟨entry-label⟩} glossaries-accsupp`

Does `⟨text⟩` with the `pluralaccess` replacement text (if set).

`\glspluralsuffix`

Expands to the letter “s” and is used to form default plurals. This command isn’t language-sensitive as there’s guarantee when it will be expanded. (It may be expanded when the entry is defined or it may be expanded when the entry is used). If you need to suppress this suffix for abbreviations, use the `noshortplural` attribute. If you need an apostrophe before the “s” for single-letter abbreviations to avoid ambiguity, use the `aposplural` attribute.

`\glspostdescription glossaries`

A hook that is usually placed after the description in glossary styles. Some of the styles provided with the `glossaries` package don’t use this hook. The `glossaries-extra-stylemods` redefines those styles to include the hook. The default definition of this command tests for the `nopostdot` option, but the `postpunc` option redefines the command to implement the chosen punctuation.

`\glspostinline glossary-inline v3.03+`
Command Summary

Used at the end of the \glossary environment.

\glspostinlinedescformat\{⟨description⟩\}\{⟨symbol⟩\}\{⟨location list⟩\}
glossary–inline v3.03+

Formats the top-level entry’s description, symbol and location list.

\glspostinlinesubdescformat\{⟨description⟩\}\{⟨symbol⟩\}\{⟨location list⟩\}
glossary–inline v3.03+

Formats the child entry’s description, symbol and location list.

\glspostlinkhook
glossaries v4.16+

A post-link hook used after all the \gls-like and \glstext-like commands. This is redefined by glossaries–extra to use \glsxtrpostlinkhook.

\glsprefixsep
initial: empty glossaries–prefix v4.45+

Separator between the prefix and the term.

\glsprestandardsort\{⟨sort cs⟩\}\{⟨type⟩\}\{⟨entry-label⟩\}
glossaries v3.13a+

Hook used with sort=standard to adjust the default sort value (with \makeglossaries or \makenoidxglossaries only).

\glsprepostlink\{⟨category⟩\}\{⟨code⟩\}
glossaries–extra v1.49+

Prepends ⟨code⟩ to post-link hook associated with the category identified by the label ⟨category⟩ (or simply defines it, if it doesn’t already exist).
## Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>\GLSp{⟨entry-label⟩}</td>
<td>Shortcut for \GLSxtrp{short}{⟨entry-label⟩}.</td>
<td>§5.4; 232</td>
</tr>
<tr>
<td>\Glsps{⟨entry-label⟩}</td>
<td>Shortcut for \Glsxtrp{short}{⟨entry-label⟩}.</td>
<td>§5.4; 232</td>
</tr>
<tr>
<td>\glsps{⟨entry-label⟩}</td>
<td>Shortcut for \Glssxtrp{short}{⟨entry-label⟩}.</td>
<td>§5.4; 232</td>
</tr>
<tr>
<td>\GLSpt{⟨entry-label⟩}</td>
<td>Shortcut for \GLSxtrp{text}{⟨entry-label⟩}.</td>
<td>§5.4; 232</td>
</tr>
<tr>
<td>\Glspst{⟨entry-label⟩}</td>
<td>Shortcut for \Glssxtrp{text}{⟨entry-label⟩}.</td>
<td>§5.4; 232</td>
</tr>
<tr>
<td>\glspt{⟨entry-label⟩}</td>
<td>Shortcut for \Glssxtrp{text}{⟨entry-label⟩}.</td>
<td>§5.4; 232</td>
</tr>
<tr>
<td>\glsrefentry{⟨entry-label⟩}</td>
<td>References (using \ref) the entry counter or sub-counter (if entrycounter or subentrycounter options are set) otherwise just does \gls{⟨entry-label⟩}.</td>
<td>§5.4; 232</td>
</tr>
</tbody>
</table>

References (using \ref) the entry counter or sub-counter (if entrycounter or subentrycounter options are set) otherwise just does \gls{⟨entry-label⟩}.

\glsrenewcommand{⟨cs⟩}[⟨n⟩][⟨default⟩]{⟨definition⟩}  
\textit{modifier: *}  
§11.5.8; 598

790
Command Summary

Like `\renewcommand` but only issues a warning instead of an error if the command hasn’t been defined.

\begin{verbatim}
\glsreset{⟨entry-label⟩}
glossaries
\end{verbatim}

Globally resets the entry’s first use flag. That is, this marks the entry as “not used”.

\begin{verbatim}
\glsresetall[⟨types⟩]
glossaries
\end{verbatim}

Globally resets all entries associated with the listed glossaries or all glossaries if ⟨types⟩ is omitted.

\begin{verbatim}
\glsresetcurrcountfalse
glossaries-extra v1.49+
\end{verbatim}

Sets \ifglsresetcurrcount to false.

\begin{verbatim}
\glsresetcurrcounttrue
glossaries-extra v1.49+
\end{verbatim}

Sets \ifglsresetcurrcount to true.

\begin{verbatim}
\glsresetentrylist
glossaries
\end{verbatim}

Inserted into the glossary code to counteract the effect of \glsnonextpages.

\begin{verbatim}
\glssee[⟨tag⟩]{⟨entry-label⟩}{⟨xr-list⟩}
glossaries v1.17+
\end{verbatim}

Indexes the entry identified by ⟨entry-label⟩ as a general cross-reference to the entries identified in the comma-separated list ⟨xr-list⟩. The optional argument is the textual tag that’s inserted before the cross-reference list and defaults to \seealso.

\begin{verbatim}
\glsseefirstitem{⟨entry-label⟩}
glossaries-extra v1.47+
\end{verbatim}

\section{Notes}

791
Command Summary

Used by \glsseelist to format the first entry.

\begin{verbatim}
\glsseeformat{⟨tag⟩}{⟨xr-list⟩}{⟨location⟩}
glossaries v1.17+
\end{verbatim}

Used to format the see cross-reference in the location list. This requires a location argument for makeindex even though it isn’t required. The default definition is \textit{⟨tag⟩} \glsseeitem{⟨xr-list⟩}.

\begin{verbatim}
\glsseeformat{⟨tag⟩}{⟨xr-list⟩}{⟨location⟩}
glossaries v1.17+
\end{verbatim}

Used by \glsseeitem to format each entry.

\begin{verbatim}
\glsseeitem{⟨entry-label⟩}
glossaries v1.17+
\end{verbatim}

Used by \glsseeitem to produce the hyperlink text.

\begin{verbatim}
\glsseeitemformat{⟨entry-label⟩}
glossaries v3.0+
\end{verbatim}

§5.13; 303

Used by \glsseeitem as a separator between penultimate and final entry in the list if there are at least three entries in the list.

\begin{verbatim}
\glsseelastoxfordsep
glossaries-extra v1.47+
\end{verbatim}

§5.13; 304

Iterates over a comma-separated list of entry labels ⟨csv-list⟩ and formats them. Each label in the list is encapsulated with \glsseeitem (or \mgsseeitem, the label corresponds to a multi-entry). The separators are \glsseelastsep (between the penultimate and last items) and \glsseesep (between all the other items). With glossaries-extra, the first label is encapsulated with \glsseefirstitem (or \mglsseefirstitem) and the final separator for a list consisting of at least three items is given by \glsseelastoxfordsep.

792
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Version Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsseesep</td>
<td>Used by \glsseelist as a separator between each entry except the last pair.</td>
<td>§5.13; 304</td>
</tr>
<tr>
<td>\glssentencecase{(text)}</td>
<td>Used by sentence case commands, such as \Gls, to perform the case change. This is simply defined to use \makefirstuc.</td>
<td>§5.2.1; 197</td>
</tr>
<tr>
<td>\glssetabbrvfmt{(category)}</td>
<td>Implements the \textit{display definitions} code for the abbreviation style associated with the given category.</td>
<td>§4.5.2; 159</td>
</tr>
<tr>
<td>\glssetattribute{(entry-label)}{(attribute)}{(value)}</td>
<td>Locally sets the given attribute to \textit{value} for the category associated with the entry identified by \textit{entry-label}.</td>
<td>§10.2.2; 529</td>
</tr>
<tr>
<td>\glssetcategoriesattribute{(category list)}{(attribute)}{(value)}</td>
<td>Globally sets the given attribute to \textit{value} for all the categories in the comma-separated list \textit{category list}.</td>
<td>§10.2.2; 529</td>
</tr>
<tr>
<td>\glssetcategoriesattributes{(category list)}{(attribute list)}{(value)}</td>
<td>Globally sets each attribute in the comma separated \textit{attribute list} to \textit{value} for each category in the comma-separated list \textit{category list}.</td>
<td>§10.2.2; 529</td>
</tr>
<tr>
<td>\glssetcategoryattribute{(category)}{(attribute)}{(value)}</td>
<td></td>
<td>§10.2.2; 529</td>
</tr>
</tbody>
</table>
Locally sets the given attribute to \langle value \rangle for the given category.

\texttt{\textbackslash glssetcategoryattributes\{\langle category \rangle\}\{\langle attribute list \rangle\}\{\langle value \rangle\}}
glossaries-extra v1.49+

§10.2.2; 529

Globally sets each attribute in the comma separated \langle attribute list \rangle to \langle value \rangle for the given \langle category \rangle.

\texttt{\textbackslash glssetcombinedsepabbrvnbsp} glossaries-extra v1.48+

§7.4; 348

Defines the multi-entry separators to use a non-breaking space (\textendash) for abbreviations.

\texttt{\textbackslash glssetcombinedsepabbrvnone} glossaries-extra v1.48+

§7.4; 348

Defines the multi-entry separators to use no separator for abbreviations.

\texttt{\textbackslash glssetcombinedsep\{\langle width \rangle\}\{\langle narrow-sep \rangle\}} glossaries-extra v1.48+

§7.4; 349

Defines the multi-entry separators to use \langle narrow-sep \rangle if the width of associated field values is less than \langle width \rangle.

\texttt{\textbackslash glssetexpandfield\{\langle field \rangle\}} glossaries v3.13a+

Expand the value of the field identified by its internal field label when defining entries (overrides \texttt{\textbackslash glsnoexpandfields}).

\texttt{\textbackslash glssetnoexpandfield\{\langle field \rangle\}} glossaries v3.13a+

Don’t expand the value of the field identified by its internal field label when defining entries (overrides \texttt{\textbackslash glsexpandfields}).
Command Summary

\glssetregularcategory{$\text{category}$}

Locally sets the regular attribute to true for the given category.

\glssetwidest[{level}]{$\text{name}$}
glossary-tree

Indicates that $\text{name}$ is the widest name for the given hierarchical level.

\glsshortaccessdisplay{$\text{text}$}{$\text{entry-label}$}
glossaries-accsupp

Does $\text{text}$ with the shortaccess replacement text (if set).

\glsshortaccsupp{$\text{replacement}$}{$\text{content}$}
glossaries-accsupp v4.45+

Applies $\text{replacement}$ as the expansion (E) attribute for $\text{content}$ using \glsaccessibility.

\glsshortpltok

§4.5.3.1; 164

A token register that stores the short plural form (which may have been modified after being passed to \newabbreviation).

\glsshortpluralaccessdisplay{$\text{text}$}{$\text{entry-label}$}
glossaries-accsupp

Does $\text{text}$ with the shortpluralaccess replacement text (if set).

\glsshorttok
glossaries

§4.5.3.1; 164

A token register that stores the short form (which may have been modified after being passed to \newabbreviation).
Command Summary

\texttt{\textbackslash gls\textbackslash show\textbackslash target\{(target-name)\}}
glossaries v4.32+

Formats the target name when \texttt{debug=showtargets} is enabled using either \texttt{\textbackslash gls\textbackslash show\textbackslash target\-inner} or \texttt{\textbackslash gls\textbackslash show\textbackslash target\-outer}, depending on the current mode.

\texttt{\textbackslash gls\textbackslash show\textbackslash target\textbackslash font}
glossaries v4.45+

Font declaration used by debugging annotations.

\texttt{\textbackslash gls\textbackslash show\textbackslash target\textbackslash font\textbackslash text\{(text)\}}
glossaries v4.50+

Text-block command that checks for math mode and switches to the font given by the \texttt{\textbackslash gls\textbackslash show\textbackslash target\textbackslash font} declaration.

\texttt{\textbackslash gls\textbackslash show\textbackslash target\inner\{(target-name)\}}
glossaries v4.50+

Formats the target name for inner and maths mode when \texttt{debug=showtargets} is enabled.

\texttt{\textbackslash gls\textbackslash show\textbackslash target\textbackslash innersym\textbackslash left\{name\}}
glossaries-extra v1.48+

§2.5; 30

Shows the left inner annotation followed by the left marker symbol \texttt{\textbackslash gls\textbackslash xtr\textbackslash show\textbackslash target\textbackslash symbol\textbackslash left}.

\texttt{\textbackslash gls\textbackslash show\textbackslash target\textbackslash innersym\textbackslash right\{name\}}
glossaries-extra v1.48+

§2.5; 30

Shows the right marker symbol \texttt{\textbackslash gls\textbackslash xtr\textbackslash show\textbackslash target\textbackslash symbol\textbackslash right} followed by the right inner annotation.

\texttt{\textbackslash gls\textbackslash show\textbackslash target\outer\{(target-name)\}}
glossaries v4.50+

Formats the target name for outer mode when \texttt{debug=showtargets} is enabled. This places a marker (\texttt{\textbackslash gls\textbackslash show\textbackslash target\symbol}) in the text and \texttt{(target-name)} in the margin.

796
Command Summary

\texttt{\textbackslash glshowtargetsymbol} \quad \text{glossaries v4.45+}

Marker (⊿) used in debugging annotations.

\texttt{\textbackslash glsstartrange}\{(\textit{options})\}\{(\textit{entry label list})\} \quad \text{glossaries-extra v1.50+} \quad \S 5.8; 262

Essentially does \texttt{\textbackslash glsaddeach}\{(\textit{options}),format=\{(\textit{encap})\}\{(\textit{entry label list})\} where \{\textit{encap}\} can either be provided by the \texttt{format} key in \{\textit{options}\} or will default to the format given in \texttt{\textbackslash GlsXtrSetDefaultRangeFormat}.

\texttt{\textbackslash glssubentryitem}\{(\textit{entry-label})\} \quad \text{glossaries v3.0+}

Does nothing if \texttt{subentrycounter=false}, otherwise increments and displays the associated counter.

\texttt{\textbackslash glssubgroupheading}\{(\textit{previous level})\}\{(\textit{level})\}\{(\textit{parent-label})\}\{(\textit{group-label})\} \quad \text{glossaries-extra v1.49+} \quad \S 8.4.1; 398

Used to format sub-group headings. Only applicable with the “unsrt” family of commands. This command won’t occur in glossaries that use \texttt{\textbackslash printglossary} or \texttt{\textbackslash printnoidxglossary}.

\texttt{\textbackslash GLSsymbol}\{(\textit{options})\}\{(\textit{entry-label})\}\{(\textit{insert})\} \quad \text{modifiers: * + \{alt-mod\}}

As \texttt{\textbackslash glssymbol} but converts the link text to all caps.

\texttt{\textbackslash Glssymbol}\{(\textit{options})\}\{(\textit{entry-label})\}\{(\textit{insert})\} \quad \text{modifiers: * + \{alt-mod\}}

As \texttt{\textbackslash glssymbol} but converts the link text to sentence case. Use \texttt{\Glossentrysymbol} within custom glossary styles instead of this command.
References the entry identified by \textit{entry-label}. The text produced is obtained from the \texttt{symbol} value. The \textit{insert} argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \texttt{\textbackslash glslink options}. Use \texttt{\textbackslash glossentrysymbol} within custom glossary styles instead of this command.

\texttt{\textbackslash glssymbolaccessdisplay\{\text{\textbackslash glosstext\}}\{\texttt{\textbackslash entry-label}\}} \texttt{\textbackslash glossaries\textbackslash accsupp}

Does \textit{text} with the \texttt{symbolaccess} replacement text (if set).

\texttt{\textbackslash glssymbolplural\{\textit{options}\}\{\textit{entry-label}\}\{\textit{insert}\}} \texttt{\textbackslash glossaries\textbackslash accsupp}

As \texttt{\textbackslash glssymbol} but for the \texttt{symbolplural} field.

\texttt{\textbackslash glssymbolpluralaccessdisplay\{\text{\textbackslash glosstext\}}\{\texttt{\textbackslash entry-label}\}} \texttt{\textbackslash glossaries\textbackslash accsupp}

Does \textit{text} with the \texttt{symbolpluralaccess} replacement text (if set).

\texttt{\textbackslash glssymbols grou pname} \texttt{\texttt{\textbackslash initial:\ Symbols \textbackslash glossaries}} (language-sensitive)

Expands to the title of the \texttt{symbols} group and (if the \texttt{symbols} package option is used) the \texttt{symbols} glossary.

\texttt{\textbackslash glistableblocksubentry\{\texttt{\textbackslash entry-label}\}} \texttt{\textbackslash glossary\textbackslash table v1.49+}

Displays the child entry identified by \textit{entry-label}. This command is redefined by block styles.


**Separator used by \glstableChildEntries between child entries.**

\glstablecaption{⟨lot title⟩}{⟨title⟩}{⟨label code⟩}

Produces the caption for the first page of the table.

\glstablecenteralign{⟨width⟩}

Expands to \texttt{c} or \texttt{p}⟨width⟩ or \texttt{\protect\centeringp}⟨width⟩, depending on the \texttt{par} setting.

\glstableChildEntries{⟨entry-label⟩}

Iterates over the \texttt{childlist} field and formats each child entry in the list for use in the block styles. Does nothing if the list is empty.

\glstableDescFmt{⟨text⟩}

Formatting applied to the description.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glistabledescheader</td>
<td>Header for the description column.</td>
<td>§8.7.4.2, 8.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>490, 499</td>
</tr>
<tr>
<td>\glistabledescwidth</td>
<td>Length register used for the width of the description column with \texttt{par=justified} or \texttt{par=ragged}. Set by the block style.</td>
<td>§8.7.4.4;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>496</td>
</tr>
<tr>
<td>\glistableHeaderFmt{\text}</td>
<td>Formats the header.</td>
<td>§8.7.4.4;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>499</td>
</tr>
<tr>
<td>\glistableiffilter{\entry-label}{\true}{\false}</td>
<td>Internally used by the custom handler in \texttt{\printunsrttable} to perform additional filtering. This command should do \texttt{\true} if the entry identified by \texttt{\entry-label} should be filtered and \texttt{\false} otherwise.</td>
<td>§8.7.4;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>484</td>
</tr>
<tr>
<td>\glistableiffilterchild{\entry-label}{\true}{\false}</td>
<td>Internally used by \texttt{\glistableChildEntries} to filter child entries. This command should do \texttt{\true} if the child entry identified by \texttt{\entry-label} should be filtered and \texttt{\false} otherwise.</td>
<td>§8.7.4.1;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>487</td>
</tr>
<tr>
<td>\glistableifhasotherfield{\entry-label}{\true}{\false}</td>
<td>Expands to \texttt{\true} if the other field is non-void for the given entry otherwise expands to \texttt{\false}.</td>
<td>§8.7.4.4;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>498</td>
</tr>
<tr>
<td>\glistableleftalign{\width}</td>
<td></td>
<td>§8.7.4.4;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>494</td>
</tr>
</tbody>
</table>

800
Expands to \l or \p{⟨width⟩} or >\protect\raggedright\p{⟨width⟩}, depending on the \texttt{par} setting.

\texttt{\glstblenamecolalign} \hfill glossary-table v1.49+ \hfill §8.7.4.4; 495

Expands to the alignment of the name column.

\texttt{\glstblenameFmt{⟨text⟩}} \hfill glossary-table v1.50+ \hfill §8.7.4.4; 497

Formatting applied to the name.

\texttt{\glstblenameheader} \hfill glossary-table v1.49+ \hfill §§8.7.4.2, 8.7.4.4; 490, 499

Header for the name column.

\texttt{\glstblenamewidth} \hfill glossary-table v1.49+ \hfill §8.7.4.4; 496

Length register used for the width of the name column with \texttt{par=justified} or \texttt{par=ragged}. Set by the block style.

\texttt{\glstblenewline} \hfill glossary-table v1.50+ \hfill §8.7.4.4; 494

Used to start a new row.

\texttt{\glstblenextcaption{⟨lot title⟩}{⟨title⟩}} \hfill glossary-table v1.49+ \hfill §8.7.4.2; 489

Produces the caption for following pages of the table.

\texttt{\glstableOther{⟨entry-label⟩}} \hfill glossary-table v1.50+ \hfill §8.7.4.4; 498

Used to display the other field.

801
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Version</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glstableothercolalign</td>
<td>Expands to the alignment of the other column.</td>
<td>§8.7.4.4; 496</td>
<td>496</td>
</tr>
<tr>
<td>\glstableotherfield</td>
<td>Expands to the internal field label of the other field.</td>
<td>§8.7.4.4; 498</td>
<td>498</td>
</tr>
<tr>
<td>\glstableotherFmt{⟨text⟩}</td>
<td>Formatting applied to the other field.</td>
<td>§8.7.4.4; 498</td>
<td>498</td>
</tr>
<tr>
<td>\glstableotherheader</td>
<td>Header for the other column.</td>
<td>§8.7.4.4; 499</td>
<td>499</td>
</tr>
<tr>
<td>\glstableotherwidth</td>
<td>Length register used for the width of the other column with \texttt{par=justified} or \texttt{par=ragged}. Set by the block style.</td>
<td>§8.7.4.4; 496</td>
<td>496</td>
</tr>
<tr>
<td>\glstablepostnextcaption</td>
<td>Appended to the caption in \glstablednextcaption.</td>
<td>§8.7.4.2; 489</td>
<td>489</td>
</tr>
<tr>
<td>\glstablepostpreambleskip</td>
<td>Length register that specifies the vertical skip after the preamble.</td>
<td>§8.7.4.4; 496</td>
<td>496</td>
</tr>
<tr>
<td>\glstablePreChildren</td>
<td></td>
<td>§8.7.4.1; 487</td>
<td>487</td>
</tr>
</tbody>
</table>

802
Command Summary

Code performed by \texttt{\glstableChildEntries} before the child list.

\texttt{\glstableprepostambleskip} \textit{initial: 5pt} \textit{glossary-table v1.50+} \texttt{§8.7.4.4; 497}

Length register that specifies the vertical skip before the postamble.

\texttt{\glstablerightalign}\{\textit{width}\} \textit{glossary-table v1.49+} \texttt{§8.7.4.4; 495}

Expands to \texttt{r} or \texttt{p}\{\textit{width}\} or >\texttt{\protect\raggedleft}\{\textit{width}\}, depending on the \texttt{par} setting.

\texttt{\glstablesetstyle}\{\textit{style-name}\} \textit{glossary-table v1.49+} \texttt{§8.7.4.3; 491}

Sets the default block style.

\texttt{\glstableSubDescFmt}\{\textit{text}\} \textit{glossary-table v1.50+} \texttt{§8.7.4.4; 498}

Formatting applied to the child description.

\texttt{\glstablesubentryalign} \textit{glossary-table v1.50+} \texttt{§8.7.4.1; 488}

Expands to the column alignment used by \texttt{\glstableSubentries}.

\texttt{\glstableSubNameFmt}\{\textit{text}\} \textit{glossary-table v1.50+} \texttt{§8.7.4.4; 497}

Formatting applied to the child name.

\texttt{\glstableSubOther}\{\textit{entry-label}\} \textit{glossary-table v1.50+} \texttt{§8.7.4.4; 498}

Used to display the sub-entry other field.
Command Summary

\glistableSubOtherFmt\{⟨text⟩\} \hspace{1cm} glossary-table v1.50+

Formatting applied to the other field.

\glistableSubSymbolFmt\{⟨text⟩\} \hspace{1cm} glossary-table v1.50+

Formatting applied to the child symbol.

\glstableSymbolcolalign \hspace{1cm} glossary-table v1.49+

Expands to the alignment of the symbol column.

\glistableSubSymbolFmt\{⟨text⟩\} \hspace{1cm} glossary-table v1.50+

Formatting applied to the symbol.

\glstableSymbolheader \hspace{1cm} glossary-table v1.49+

Header for the symbol column.

\glstableSymbolwidth \hspace{1cm} glossary-table v1.49+

Length register used for the width of the symbol column with \texttt{par=justified} or \texttt{par=ragged}. Set by the block style.

\glstarget\{⟨entry-label⟩\}\{⟨text⟩\} \hspace{1cm} glossaries v1.18+

Used by glossary styles to create a hypertarget (if enabled) for the entry (identified by \texttt{⟨entry-label⟩}). The \texttt{(text)} is usually \texttt{\glossentryname\{⟨entry-label⟩\}}, but it can be something else.

\GLStext\{⟨options⟩\}\{⟨entry-label⟩\}\{⟨insert⟩\} \hspace{1cm} modifiers: * + \texttt{⟨alt-mod⟩} glossaries

804
Command Summary

As \texttt{\textbackslash glstext} but converts the link text to all caps. If you have defined the entry with \texttt{\textbackslash new-abbreviation} use \texttt{\textbackslash GLSxtrshort} or \texttt{\textbackslash GLS[preunset]} instead.

\begin{verbatim}
\Glstext[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩ glossaries
\end{verbatim}

As \texttt{\textbackslash glstext} but converts the first character of the link text to uppercase (for the start of a sentence) using \texttt{\textbackslash makefirstuc}. If you have defined the entry with \texttt{\textbackslash newabbreviation} use \texttt{\textbackslash Glsxtrshort} or \texttt{\textbackslash Gls[preunset]} instead.

\begin{verbatim}
\glstext[⟨options⟩]{⟨entry-label⟩}[⟨insert⟩] modifiers: * + ⟨alt-mod⟩ glossaries
\end{verbatim}

References the entry identified by \texttt{⟨entry-label⟩}. The text produced is obtained from the \texttt{text} value. The \texttt{⟨insert⟩} argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. If you have defined the entry with \texttt{\textbackslash newabbreviation} use \texttt{\textbackslash glsxtrshort} for the short form or \texttt{\textbackslash gls[preunset]}, as some abbreviation styles are too complicated to work with \texttt{\textbackslash glstext}. For the first optional argument, see \texttt{\textbackslash glslink} options.

\begin{verbatim}
\glstextaccessdisplay{⟨text⟩}{⟨entry-label⟩} glossaries-accsupp
\end{verbatim}

Does \texttt{⟨text⟩} with the \texttt{textaccess} replacement text (if set).

\begin{verbatim}
\glstextformat{⟨text⟩} glossaries v1.04+
\end{verbatim}

The default outer text formatting command used by the \texttt{\textbackslash gls-like} and \texttt{\textbackslash glstext-like} commands.

\begin{verbatim}
\glstextup{⟨text⟩} glossaries v3.09a+
\end{verbatim}

Counteracts the effect of \texttt{\textbackslash textsc}.

\begin{verbatim}
\glstildechar glossaries v4.10+
\end{verbatim}

805
Command Summary

Expands to a literal tilde character.

\texttt{\textbackslash glstopicAssignSubIndent}\{\langle level\rangle\}  
\textit{glossary-topic v1.40+}  
§8.7.3; 481

Used to set the indentation for sub-levels.

\texttt{\textbackslash glstopicAssignWidest}\{\langle level\rangle\}  
\textit{glossary-topic v1.40+}  
§8.7.3; 481

Used by \texttt{\textbackslash glstopicAssignSubIndent} to calculate the width of the widest name for the given level.

\texttt{\textbackslash glstopicCols}  
\textit{glossary-topic v1.40+}  
§8.7.3; 477

Expands to the number of columns for \texttt{topicmcols}.

\texttt{\textbackslash glstopicColsEnv}  
\textit{glossary-topic v1.40+}  
§8.7.3; 476

Expands to the \texttt{multicols} environment name to use for \texttt{topicmcols}.

\texttt{\textbackslash glstopicDesc}\{\langle entry-label\rangle\}  
\textit{glossary-topic v1.40+}  
§8.7.3; 481

Used to format the top-level description.

\texttt{\textbackslash glstopicGroupHeading}\{\langle group-label\rangle\}  
\textit{glossary-topic v1.40+}  
§8.7.3; 479

Used to format the top-level group headings, if required.

\texttt{\textbackslash glstopicInit}  
\textit{glossary-topic v1.40+}  
§8.7.3; 479

Initialisation hook.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glstopicItem{⟨entry-label⟩}{⟨location list⟩}</td>
<td>Used to format top-level entries.</td>
</tr>
<tr>
<td>\glstopicLoc{⟨entry-label⟩}{⟨location list⟩}</td>
<td>Used to format the top-level location list.</td>
</tr>
<tr>
<td>\glstopicMarker{⟨entry-label⟩}</td>
<td>Hook inserted before a top-level entry.</td>
</tr>
<tr>
<td>\glstopicMidSkip</td>
<td>Vertical space inserted before the description for a top-level entry.</td>
</tr>
<tr>
<td>\glstopicParIndent</td>
<td>Length register used for the top-level paragraph indent.</td>
</tr>
<tr>
<td>\glstopicPostSkip</td>
<td>Vertical space inserted after the description for a top-level entry.</td>
</tr>
<tr>
<td>\glstopicPreSkip</td>
<td>Vertical space inserted before a top-level entry.</td>
</tr>
<tr>
<td>\glstopicSubGroupHeading{(prev group level)}{(group level)}{⟨parent entry⟩}{⟨group-label⟩}</td>
<td></td>
</tr>
</tbody>
</table>
Command Summary

Used to format the sub-group headings, if supported.

\glstopicSubIndent \hspace{0.5cm} glossary-topic v1.40+  
$\S$8.7.3; 479

Length register used for the child indent.

\glstopicSubItem{⟨level⟩}{⟨entry-label⟩}{⟨location list⟩} \hspace{0.5cm} glossary-topic v1.40+  
$\S$8.7.3; 481

Used to format child entries.

\glstopicSubItemBox{⟨level⟩}{⟨text⟩} \hspace{0.5cm} glossary-topic v1.40+  
$\S$8.7.3; 482

Horizontal box used for child name if a widest name has been provided.

\glstopicSubItemParIndent \hspace{0.5cm} glossary-topic v1.46+  
$\S$8.7.3; 479

Length register used for the child paragraph indent.

\glstopicSubItemSep \hspace{0.5cm} glossary-topic v1.40+  
$\S$8.7.3; 482

Horizontal separator used after child names.

\glstopicSubLoc{⟨entry-label⟩}{⟨location list⟩} \hspace{0.5cm} glossary-topic v1.41+  
$\S$8.7.3; 482

Formats the child location lists.

\glstopicSubNameFont{⟨text⟩} \hspace{0.5cm} glossary-topic v1.40+  
$\S$8.7.3; 481

Font command to apply to the child entry name.
### Command Summary

- **\glstopicSubPreLocSep**
  - Separator before the child location lists.
  - Glossary Topic v1.41+
  - §8.7.3; 482

- **\glstopicTitle\{⟨entry-label⟩\}**
  - Used to format the name and (if provided) symbol for the top-level entry title.
  - Glossary Topic v1.40+
  - §8.7.3; 480

- **\glstopicTitleFont\{⟨text⟩\}**
  - Font command to apply to the top-level entry title.
  - Glossary Topic v1.40+
  - §8.7.3; 480

- **\glstreechilddesc\{⟨entry-label⟩\}**
  - Displays the given child entry’s description with pre and post hooks for the tree styles.
  - Glossaries-Extra-Stylemods v1.31+
  - §8.6.5.4; 438

- **\glstreeChildDescLoc\{⟨entry-label⟩\}\{⟨location list⟩\}**
  - Formats the child description (if set) and location list for the tree styles.
  - Glossaries-Extra-Stylemods v1.41+
  - §8.6.5.4; 439

- **\glstreechildpredesc**
  - Space inserted before child descriptions.
  - Glossary Tree v4.26+
  - §8.6.5.4; 437

- **\glstreechildprelocation**
  - Used before the child entry location list for the tree and index styles.
  - Glossaries-Extra-Stylemods v1.21+
  - §8.6.5.4; 437

- **\glstreechildsymbol\{⟨entry-label⟩\}**
  - Glossaries-Extra-Stylemods v1.31+
  - §8.6.5.4; 438
Displays the top-level symbol in parentheses, if set, for the tree styles.

\glstreedefaultnamefmt\{⟨text⟩\} glossaries-extra-stylemods v1.31+

Used as the default name format for the tree and index styles.

\glstreedesc\{⟨entry-label⟩\} glossaries-extra-stylemods v1.31+

Displays the given top-level entry’s description with pre and post hooks for the tree styles.

\glstreeDescLoc\{⟨entry-label⟩\}\{⟨location list⟩\} glossaries-extra-stylemods v1.41+

Formats the top-level description (if set) and location list for the tree styles.

\glstreegroupheaderfmt\{⟨text⟩\} glossary-tree v4.22+

Used to format the group title for the treegroup and indexgroup styles.

\glstreegroupheaderskip glossaries-extra-stylemods v1.42+

After group header skip for the treegroup and indexgroup styles.

\glstreegroupskip glossaries-extra-stylemods v1.41+

Group skip for the tree and index styles.

\glstreeitem glossary-tree v4.26+

Used to indent the top-level entries for the index styles.
Command Summary

\texttt{\textbackslash glistrenamefmt\{⟨text⟩\}} \hspace{1cm} glossary-tree v4.08+

Used to format the name for the tree and index styles.

\texttt{\textbackslash glistreenavigationfmt\{⟨text⟩\}} \hspace{1cm} glossary-tree v4.22+

Used to format the navigation element for styles like treehypergroup.

\texttt{\textbackslash glistreeNoDescSymbolPreLocation} \hspace{1cm} glossaries-extra-stylemods v1.42+

Inserted before the location list when there’s no description or symbol for the tree styles.

\texttt{\textbackslash glistreenonamechilddesc\{⟨entry-label⟩\}} \hspace{1cm} glossaries-extra-stylemods v1.31+

Displays the given child entry’s description and post hook for the treenoname styles.

\texttt{\textbackslash glistreenonameChildDescLoc\{⟨entry-label⟩\}} \hspace{1cm} glossaries-extra-stylemods v1.45+

Displays the given child entry’s description and location list for the treenoname styles.

\texttt{\textbackslash glistreenamedesc\{⟨entry-label⟩\}} \hspace{1cm} glossaries-extra-stylemods v1.31+

Displays the given top-level entry’s description with pre and post hooks for the treenoname styles.

\texttt{\textbackslash glistreenamedescLoc\{⟨entry-label⟩\}} \hspace{1cm} glossaries-extra-stylemods v1.45+

Displays the given top-level entry’s description and location list for the treenoname styles.

\texttt{\textbackslash glistreenamesymbol\{⟨entry-label⟩\}} \hspace{1cm} glossaries-extra-stylemods v1.31+

§8.6.5.4; 436

§8.6.5.4; 439

§8.6.5.4; 438

§8.6.5.4; 438

§8.6.5.4; 438

811
Displays the given top-level entry’s symbol in parentheses for the treenoname styles.

\texttt{\glstreepredesc} \hspace{2cm} \textit{glossary-tree v4.26+}

Space inserted before top-level descriptions.

\texttt{\glstreePreHeader\{\langle group-label\rangle\}\{\langle group-title\rangle\} \hspace{2cm} \textit{glossaries-extra-stylemods v1.41+}}

Pre group header hook the treegroup and indexgroup styles.

\texttt{\glstreeprelocation \hspace{2cm} \textit{initial: \glxsotrprelocation}} \hspace{2cm} \textit{glossaries-extra-stylemods v1.21+}

Used before the top-level entry location list for the tree and index styles.

\texttt{\glstreesubgroupitem\{previous group level\}\{level\}\{parent label\}\{group label\}\{group title\} \hspace{2cm} \textit{glossaries-extra-stylemods v1.49+}}

Used to display the sub-group header in the treegroup styles.

\texttt{\glstreesubitem} \hspace{2cm} \textit{glossary-tree v4.26+}

Used to indent the level 1 entries for the index styles.

\texttt{\glstreeSubPreHeader\{\langle previous group level\rangle\}\{\langle level\rangle\}\{\langle parent label\rangle\}\{\langle group label\rangle\}\{\langle group title\rangle\} \hspace{2cm} \textit{glossaries-extra-stylemods v1.49+}}

Pre sub-group header hook the treegroup and indexgroup styles.

\texttt{\glstreesubsubitem} \hspace{2cm} \textit{glossary-tree v4.26+}
Used to indent the level 2 entries for the index styles.

\glstreesymbol{⟨entry-label⟩}
glossaries-extra-stylemods v1.31+

§8.6.5.4; 438

Displays the top-level symbol in parentheses, if set, for the tree styles.

\glstriggerrecordformat{⟨location⟩}
glossaries-extra v1.21+

§11.4; 550

Used as a special location format that indicates that the record is a trigger record.

\glsunexpandedfieldvalue{⟨entry-label⟩}{⟨field-label⟩}
glossaries v4.48+

For use in expandable contexts where the field value is required but the contents should not be expanded. The field should be identified by its internal field label. Expands to nothing with no error or warning if the entry or field aren’t defined.

\glsunset{⟨entry-label⟩}
glossaries

Globally unsets the entry’s first use flag. That is, this marks the entry as “used”.

\glsunsetall[⟨types⟩]
glossaries

Globally unsets all entries associated with the listed glossaries or all glossaries if ⟨types⟩ is omitted.

\glsunsetcategoryattribute{⟨category⟩}{⟨attribute⟩}
glossaries-extra v1.47+

§10.2.2; 529

Locally unsets the given attribute for the given category.

\glsupdatewidest[⟨level⟩]{⟨name⟩}
glossaries-extra-stylemods v1.23+

§8.6.5.4; 440

813
Similar to \glssetwidest but only if \( \text{name} \) is wider than the current widest value for the given hierarchical level.

\begin{verbatim}
\glsuppercase{(text)}
glossaries v4.50+
\end{verbatim}

Converts \( \text{text} \) to uppercase.

\begin{verbatim}
\glsuseabbrvfont{(style-name)}{(text)}
glossaries-extra v1.21+
\end{verbatim}

Formats \( \text{text} \) according to the short format for the given abbreviation style.

\begin{verbatim}
\glsuselongfont{(style-name)}{(text)}
glossaries-extra v1.21+
\end{verbatim}

Formats \( \text{text} \) according to the long format for the given abbreviation style.

\begin{verbatim}
\glsuserdescription{(text)}{(entry-label)}
glossaries-extra v1.30+
\end{verbatim}

Description encapsulator for styles like long-short-user.

\begin{verbatim}
\GLSuseri[(options)]{(entry-label)}[(insert)] modifiers: * + (alt-mod)
glossaries v2.04+
\end{verbatim}

As \glsuseri but converts the link text to all caps.

\begin{verbatim}
\Glsuseri[(options)]{(entry-label)}[(insert)] modifiers: * + (alt-mod)
glossaries v2.04+
\end{verbatim}

As \glsuseri but converts the link text to sentence case.
References the entry identified by \langle entry-label \rangle. The text produced is obtained from the user1 value. The \langle insert \rangle argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options.

\glsuseriaccessdisplay{⟨text⟩}{⟨entry-label⟩} glossaries-accsupp v4.45+

Does \langle text \rangle with the user1access replacement text (if set).

\GLSuserii[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} glossaries v2.04+

As \glsuserii but converts the link text to all caps.

\Glsuseri[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} glossaries v2.04+

As \glsuseri but converts the link text to sentence case.

\glsuserii[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} glossaries v2.04+

References the entry identified by \langle entry-label \rangle. The text produced is obtained from the user2 value. The \langle insert \rangle argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options.

\glsuseriiaccessdisplay{⟨text⟩}{⟨entry-label⟩} glossaries-accsupp v4.45+

Does \langle text \rangle with the user2access replacement text (if set).

\GLSuserii[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩} glossaries v2.04+
As `\glsuseriii` but converts the link text to all caps.

```
\Glsuseriii[(options)]{(entry-label)}[(insert)]
glossaries v2.04+
```

As `\glsuseriii` but converts the link text to sentence case.

```
\glsuseriii[(options)]{(entry-label)}[(insert)]
glossaries v2.04+
```

References the entry identified by `⟨entry-label⟩`. The text produced is obtained from the `user3` value. The `⟨insert⟩` argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see `\glink` options.

```
\glsuseriiiaccessdisplay{(text)}{(entry-label)}
glossaries-accsupp v4.45+
```

Does `⟨text⟩` with the `user3access` replacement text (if set).

```
\GLSuseriv[(options)]{(entry-label)}[(insert)]
glossaries v2.04+
```

As `\glsuseriv` but converts the link text to all caps.

```
\Glsuseriv[(options)]{(entry-label)}[(insert)]
glossaries v2.04+
```

As `\glsuseriv` but converts the link text to sentence case.

```
\glsuseriv[(options)]{(entry-label)}[(insert)]
glossaries v2.04+
```
Command Summary

References the entry identified by \textit{(entry-label)}. The text produced is obtained from the \texttt{user4} value. The \textit{(insert)} argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \texttt{\glslink} options.

\begin{verbatim}
\glsuserivaccessdisplay\{\textit{text}\}\{\textit{entry-label}\} \quad \texttt{glossaries-accsupp v4.45+}
\end{verbatim}

Does \textit{text} with the \texttt{\user4access} replacement text (if set).

\begin{verbatim}
\GLSuserv\left[\texttt{\textit{options}}\right]\{\textit{entry-label}\}[(\textit{insert})] \quad \text{modifiers: * + (alt-mod)}
glossaries v2.04+
\end{verbatim}

As \texttt{\glsuserv} but converts the link text to all caps.

\begin{verbatim}
\GLSuserv\left[\texttt{\textit{options}}\right]\{\textit{entry-label}\}[(\textit{insert})] \quad \text{modifiers: * + (alt-mod)}
glossaries v2.04+
\end{verbatim}

As \texttt{\glsuserv} but converts the link text to sentence case.

\begin{verbatim}
\GLSuserv\left[\texttt{\textit{options}}\right]\{\textit{entry-label}\}[(\textit{insert})] \quad \text{modifiers: * + (alt-mod)}
glossaries v2.04+
\end{verbatim}

References the entry identified by \textit{(entry-label)}. The text produced is obtained from the \texttt{user5} value. The \textit{(insert)} argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \texttt{\glslink} options.

\begin{verbatim}
\glsuserivaccessdisplay\{\textit{text}\}\{\textit{entry-label}\} \quad \texttt{glossaries-accsupp v4.45+}
\end{verbatim}

Does \textit{text} with the \texttt{\user5access} replacement text (if set).

\begin{verbatim}
\GLSuservi\left[\texttt{\textit{options}}\right]\{\textit{entry-label}\}[(\textit{insert})] \quad \text{modifiers: * + (alt-mod)}
glossaries v2.04+
\end{verbatim}
Command Summary

As \glsuservi but converts the link text to all caps.

\Glsuservi[(options)]{(entry-label)}{(insert)} \hfill modifiers: * + (alt-mod)

As \glsuservi but converts the link text to sentence case.

\glsuservi[(options)]{(entry-label)}{(insert)} \hfill modifiers: * + (alt-mod)

References the entry identified by (entry-label). The text produced is obtained from the user6 value. The (insert) argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options.

\glsuserviaccessdisplay{(text)}{(entry-label)} \hfill glossaries-accsupp v4.45+

Does (text) with the user6access replacement text (if set).

\glswriteentry{(entry-label)}{(code)} \hfill §5.8; 268

Performs the indexing code unless indexing should be suppressed.

\glsxpabbrvfont{(text)}{(category)} \hfill glossaries-extra v1.49+

If the markshortwords attribute is set for the given category, this encapsulates (text) with \glsabbrvfont otherwise with \glsinnerfmtabbrvfont. This command has to expand, so protect any content that shouldn’t expand.

\glsxplongfont{(category)}{(text)} \hfill §4.5.3.1; 168

818
If the \texttt{markwords} attribute is set for the given category, this encapsulates \langle text \rangle with \texttt{\glslongfont} otherwise with \texttt{\glsinnerfmtlongfont}. This command has to expand, so protect any content that shouldn’t expand.

\textbf{Glsxtr}

\begin{verbatim}
\Glsxtr[(\gls-options)] [(\dfn-options)]{\langle entry-label\rangle}
\end{verbatim}

As \texttt{\glsxtr} but applies sentence case.

\begin{verbatim}
\Glsxtr[(\gls-options)] [(\dfn-options)]{\langle entry-label\rangle}
\end{verbatim}

If \langle entry-label \rangle has already been defined, this just references it, otherwise the entry is defined. This command must be enabled with \texttt{\GlsXtrEnableOnTheFly}.

\begin{verbatim}
\glsxtr@counterrecord{\langle entry-label\rangle}\{(\counter)\}(\langle value\rangle)\ glossaries-extra v1.12+
\end{verbatim}

An aux file command used with \texttt{\GlsXtrRecordCounter} to append \langle value \rangle to the \texttt{record} \langle counter \rangle field. Also implements \texttt{\glsxtrAddCounterRecordHook}.

\begin{verbatim}
\glsxtr@record{\langle entry-label\rangle}\{(\h-prefix)\}{\langle counter\rangle}\{\langle encap\rangle\}\{\langle location\rangle\}
glossaries-extra v1.08+
\end{verbatim}

Used in the aux file to provide the record for \texttt{bib2gls} (\texttt{record=only}). Ignored by \LaTeX{}.

\begin{verbatim}
\glsxtr@record@nameref{\langle entry-label\rangle}\{(\h-prefix)\}{\langle counter\rangle}\{\langle encap\rangle\}\{\langle location\rangle\}\{\langle current title\rangle\}\{\langle current anchor\rangle\}\{\langle the-h-counter\rangle\} glossaries-extra v1.37+
\end{verbatim}

Used in the aux file to provide the \texttt{nameref} record for \texttt{bib2gls}. Ignored by \LaTeX{}.

\begin{verbatim}
\glsxtr@resource{\langle options\rangle}{\langle basename\rangle}
glossaries-extra v1.08+
\end{verbatim}
## Command Summary

Used in the aux file to provide the resource options for `bib2gls` for each resource set. Ignored by `EPiX`.

\[\text{Command: } \text{	extbackslash glsxtrabbreviationfont}{\langle text\rangle}\]  
*glossaries-extra v1.30+*  

§5.5.2; 241

Used by `\glslentryfmt` to encapsulate non-regular entries the have the short field set.

\[\text{Command: } \text{	extbackslash glsxtrabbrvfootnote}{\langle entry-label\rangle}{\langle text\rangle}\]  
*glossaries-extra v1.07+*  

145

Command that produces the footnote for the footnote abbreviation styles, such as `short-footnote` and `short-postfootnote`.

\[\text{Command: } \text{	extbackslash glsxtrabbrvpluralsuffix}\]  
*initial: \text{\textbackslash glspluralsuffix}*  
*glossaries-extra v1.12+*  

§4.1.2; 44

The default plural suffix used for abbreviations.

\[\text{Command: } \text{\textbackslash glsxtrabbrvtype}\]  
*initial: \text{\textbackslash glsdefaulttype}*  
*glossaries-extra*  

§4.1.4; 44

Expands to the label of the default abbreviation glossary. The `abbreviations` package option will redefine this to `abbreviations`.

\[\text{Command: } \text{\textbackslash glsxtrAccSuppAbbrSetFirstLongAttrs\{}\langle category\rangle\}\]  
*glossaries-extra v1.42+*  

§4.5.3.1; 169

Initialised accessibility support for the `name`, `text` and `plural` fields (if enabled with `accsupp`). This command is provided for abbreviation styles where the `name` and `text` are just the formatted abbreviation.

\[\text{Command: } \text{\textbackslash glsxtrAccSuppAbbrSetNameLongAttrs\{}\langle category\rangle\}\]  
*glossaries-extra v1.42+*  

§4.5.3.1; 169

Initialised accessibility support for the `first`, `firstplural`, `text` and `plural` fields (if enabled with `accsupp`). This command is provided for abbreviation styles where the `first` and `text` are just the formatted abbreviation.
### Command Summary

**\glsxtrAccSuppAbbrSetNameShortAttrs\{(category)\}**

Initialised accessibility support for the name field (if enabled with accsupp). This command is provided for abbreviation styles where only the name is just the formatted abbreviation.

**\glsxtrAccSuppAbbrSetNoLongAttrs\{(category)\}**

Initialised accessibility support for the name, first, firstplural, text and plural fields (if enabled with accsupp). This command is provided for abbreviation styles where the name, first and text are just the formatted abbreviation.

**\glsxtrAccSuppAbbrSetTextShortAttrs\{(category)\}**

Initialised accessibility support for the text and plural fields (if enabled with accsupp). This command is provided for abbreviation styles where the text is just the formatted abbreviation.

**\glsxtractivatenopost**

Activates nopostdesc and glsxtrnopostpunc.

**\glsxtransualanchor**

Expands to the anchor required by glsxtrdisplaylocnameref.

**\glsxtraddallcrossrefs**

Iterates over all defined entries and indexes any cross-references (identified by the see or seealso keys) that haven’t been used.
User-level hook used by \glsxtr@counterrecord. If this command is redefined, it must be done so in the preamble before the aux file is input.

\glsxtraddgroup{⟨entry-label⟩}{⟨code⟩} glossaries-extra v1.49+

§8.4.1; 395

Used by the “unsrt” family of commands to perform ⟨code⟩ if the entry identified by ⟨entry-label⟩ should have support for groups.

\glsxtraddlabelprefix{⟨label-prefix⟩} glossaries-extra-bib2gls v1.37+

§11.5.7; 589

Appends ⟨label-prefix⟩ to the list of known labels.

\glsxtraddpunctuationmark{⟨token(s)⟩}

§5.5.4; 250

Adds ⟨token(s)⟩ to the list of punctuation characters used by \glsxtrifnextpunc. You may list multiple characters at the same time to add a batch, but don’t add any separators (including spaces). Note that each character must be a single token, which means a single-byte character for pdfTEX. Multi-byte characters (UTF-8) will required a native Unicode engine (XƎLATEX or LuaLaTEX).

\glsxtraddunusedxrefs glossaries-extra v1.49+

§5.9.3; 282

Indexes any cross-references (identified by the see or seealso keys) that haven’t been used.

\glsxtralias{⟨entry-label⟩} glossaries-extra v1.12+

§5.9.2; 280

Expands to the value of the alias field for the entry identified by ⟨entry-label⟩. If the field isn’t set, this will expand to nothing. If the entry isn’t defined, this will expand to \relax.

\glsxtraliashook{⟨entry-label⟩} glossaries-extra v1.49+

§3.4; 36

Hook implemented when the alias key is provided when an entry is defined.
Command Summary

\textbf{\texttt{\glsxtrAltTreeIndent}} \hspace{1cm} glossaries-extra-stylemods v1.05+

§8.6.5.4; 443

Length register for the subsequent paragraph indentation for the alttree-like styles.

\textbf{\texttt{\glsxtralttreeInit}} \hspace{1cm} glossaries-extra-stylemods v1.05+

§8.6.5.4; 442

Initialisation code performed by the alttree-like styles.

\textbf{\texttt{\glsxtralttreeSubSymbolDescLocation\{\langle entry-label\rangle\}\{\langle location list\rangle\}}}

§8.6.5.4; 442

glossaries-extra-stylemods v1.05+

Formats the symbol, description and location for child entries for the alttree-like styles.

\textbf{\texttt{\glsxtralttreeSymbolDescLocation\{\langle entry-label\rangle\}\{\langle location list\rangle\}}}

§8.6.5.4; 442

glossaries-extra-stylemods v1.05+

Formats the symbol, description and location for top-level entries for the alttree-like styles.

\textbf{\texttt{\glsxtrapptocsvfield\{\langle entry-label\rangle\}\{\langle field-label\rangle\}\{\langle element\rangle\}}}

§3.5; 38

glossaries-extra v1.47+

For use with fields that should contain comma-separated lists, this will append a command followed by \langle element\rangle to the field value. If the field isn’t defined, this command will behave like \glsxtrdeffield. No existence check is performed.

\textbf{\texttt{\GlsXtrAppToDefaultGlsOpts\{\langle options\rangle\}}}

§5.1.1; 187

glossaries-extra v1.49+

Locally append \langle options\rangle to the default options for the \gls-like and \glstext-like commands.

\textbf{\texttt{\glsxtrassignactualsetup}} \hspace{1cm} glossaries-extra v1.42+

§9.1; 500

(requires accsupp)
Command Summary

Used to strip common formatting commands from a field value to supply the text-only accessibility content when initialising the default `shortaccess` and `shortpluralaccess` values.

\[\texttt{\textbackslash glsxtrassignfieldfont}\{\textit{entry-label}\}\]  
\texttt{glossaries-extra v1.04+}  
§5.5.2; 241

Used by the \texttt{\textbackslash glstext}-like commands to initialise the formatting commands required for the given entry.

\[\texttt{\textbackslash glsxtrassignlinktextfmt}\]  
\texttt{glossaries-extra v1.49+}  
§5.5.4; 255

Initialised by the \texttt{\textbackslash gls}-like and \texttt{\textbackslash glstext}-like commands, this contains the definitions of \texttt{\textbackslash glslabel}, \texttt{\textbackslash glstextformat}, \texttt{\textbackslash glsxtrgenentrytextfmt}.

\[\texttt{\textbackslash glsxtrattentrytextfmt}\{\textit{text}\}\]  
\texttt{glossaries-extra v1.49+}  
§5.5.3; 244

Applies the command obtained from the control sequence name supplied in the `innertext-format` attribute for the category assigned to the entry given by \texttt{\textbackslash glslabel}. This command isn’t used by default as it should rarely be needed an increases complexity.

\[\texttt{\textbackslash GlsXtrAutoAddOnFormat}\{\textit{entry-label}\}\{\textit{format list}\}\{\textit{glsadd options}\}\]  
\texttt{glossaries-extra v1.37+}  
§5.8; 263

Identifies formats that should trigger an automatic \texttt{\textbackslash glsadd} by the \texttt{\textbackslash gls}-like and \texttt{\textbackslash glstext}-like commands.

\[\texttt{\textbackslash glsxtrautoindex}\]  
\texttt{initial: \textbackslash index}  
\texttt{glossaries-extra v1.16+}  
§12; 601

The indexing command used by by the auto-indexing feature.

\[\texttt{\textbackslash glsxtrautoindexasssignsort}\{\textit{entry-label}\}\]  
\texttt{glossaries-extra v1.16+}  
§12; 601

Used to assign the sort value for the auto-indexing feature.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsxtrautoindexentry{⟨entry-label⟩}</td>
<td>Expands to the “actual” part for the auto-indexing feature.</td>
<td>glossaries-extra v1.16+</td>
</tr>
<tr>
<td>\glsxtrautoindexesc</td>
<td>Escapes the sort value used by the auto-indexing feature.</td>
<td>glossaries-extra v1.36+</td>
</tr>
<tr>
<td>\glsxtrBasicDigitrules</td>
<td>Expands to the Basic Latin digit character sort rules.</td>
<td>glossaries-extra-bib2gls v1.27+</td>
</tr>
<tr>
<td>\glsxtrbibaddress{⟨entry-label⟩}</td>
<td>Accesses the address field.</td>
<td>(§11.5.2; 562)</td>
</tr>
<tr>
<td>\glsxtrbibauthor{⟨entry-label⟩}</td>
<td>Accesses the author field.</td>
<td>(§11.5.2; 562)</td>
</tr>
<tr>
<td>\glsxtrbibbooktitle{⟨entry-label⟩}</td>
<td>Accesses the booktitle field.</td>
<td>(§11.5.2; 562)</td>
</tr>
<tr>
<td>\glsxtrbibchapter{⟨entry-label⟩}</td>
<td>Accesses the chapter field.</td>
<td>(§11.5.2; 562)</td>
</tr>
</tbody>
</table>

825
Accesses the **edition** field.

\texttt{\glsxtrbibhowpublished{⟨entry-label⟩}} \hspace{1cm} \text{(defined by \GlsXtrProvideBibTeXFields)} \hspace{1cm} \S\text{11.5.2; 562}

Accesses the **howpublished** field.

\texttt{\glsxtrbibinstitution{⟨entry-label⟩}} \hspace{1cm} \text{(defined by \GlsXtrProvideBibTeXFields)} \hspace{1cm} \S\text{11.5.2; 562}

Accesses the **institution** field.

\texttt{\glsxtrbibjournal{⟨entry-label⟩}} \hspace{1cm} \text{(defined by \GlsXtrProvideBibTeXFields)} \hspace{1cm} \S\text{11.5.2; 562}

Accesses the **journal** field.

\texttt{\glsxtrbibmonth{⟨entry-label⟩}} \hspace{1cm} \text{(defined by \GlsXtrProvideBibTeXFields)} \hspace{1cm} \S\text{11.5.2; 562}

Accesses the **month** field.

\texttt{\glsxtrbibnote{⟨entry-label⟩}} \hspace{1cm} \text{(defined by \GlsXtrProvideBibTeXFields)} \hspace{1cm} \S\text{11.5.2; 562}

Accesses the **note** field.

\texttt{\glsxtrbibnumber{⟨entry-label⟩}} \hspace{1cm} \text{(defined by \GlsXtrProvideBibTeXFields)} \hspace{1cm} \S\text{11.5.2; 562}

Accesses the **number** field.

\texttt{\glsxtrbiborganization{⟨entry-label⟩}} \hspace{1cm} \text{(defined by \GlsXtrProvideBibTeXFields)} \hspace{1cm} \S\text{11.5.2; 562}

Accesses the **organization** field.

826
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\glsxtrbibpages{⟨entry-label⟩}</code></td>
<td>(defined by \GlsXtrProvideBibTeXFields) Accesses the <code>pages</code> field.</td>
</tr>
<tr>
<td><code>\glsxtrbibpublisher{⟨entry-label⟩}</code></td>
<td>(defined by \GlsXtrProvideBibTeXFields) Accesses the <code>publisher</code> field.</td>
</tr>
<tr>
<td><code>\glsxtrbschool{⟨entry-label⟩}</code></td>
<td>(defined by \GlsXtrProvideBibTeXFields) Accesses the <code>school</code> field.</td>
</tr>
<tr>
<td><code>\glsxtrbibseries{⟨entry-label⟩}</code></td>
<td>(defined by \GlsXtrProvideBibTeXFields) Accesses the <code>series</code> field.</td>
</tr>
<tr>
<td><code>\GlsXtrBibTeXEntryAliases</code></td>
<td>Glossaries-extra-bib2gls v1.29+ Expands to the \BibTeX to \bib2gls entry aliases for use in <code>entry-type-aliases</code>.</td>
</tr>
<tr>
<td><code>\glsxtrbibtitle{⟨entry-label⟩}</code></td>
<td>(defined by \GlsXtrProvideBibTeXFields) Accesses the <code>title</code> field.</td>
</tr>
<tr>
<td><code>\glsxtrbibtype{⟨entry-label⟩}</code></td>
<td>(defined by \GlsXtrProvideBibTeXFields) Accesses the <code>bibtextype</code> field.</td>
</tr>
<tr>
<td><code>\glsxtrbibvolume{⟨entry-label⟩}</code></td>
<td>(defined by \GlsXtrProvideBibTeXFields)</td>
</tr>
</tbody>
</table>

827
**Command Summary**

Accesses the `volume` field.

\begin{quote}
\verb|\glsxtrbookindexatendgroup{⟨entry-label⟩}| \quad glossary-bookindex v1.21+ \hfill §8.7.1; 447
\end{quote}

Used by the bookindex style at the end of a letter group (where the last top-level entry is given by `⟨entry-label⟩`).

\begin{quote}
\verb|\glsxtrbookindexbetween{⟨entry1-label⟩}{⟨entry2-label⟩}| \quad glossary-bookindex v1.21+ \hfill §8.7.1; 446
\end{quote}

Used by the bookindex style between two entries where `⟨entry1-label⟩` is the last top-level entry and `⟨entry2-label⟩` is the next entry, which is a top-level entry.

\begin{quote}
\verb|\glsxtrbookindexbookmark{⟨group-title⟩}{⟨bookmark-name⟩}| \quad glossary-bookindex v1.21+ \hfill §8.7.1; 448
\end{quote}

Adds a bookmark with `\pdfbookmark`, if supported.

\begin{quote}
\verb|\glsxtrbookindexcols| \quad initial: 2 \quad glossary-bookindex v1.21+ \hfill §8.7.1; 443
\end{quote}

Expands to the number of columns for the bookindex style.

\begin{quote}
\verb|\glsxtrbookindexcolspread| \quad glossary-bookindex v1.12+ \hfill §8.7.1; 444
\end{quote}

If not empty this should expand to the option argument for `multicols`.

\begin{quote}
\verb|\glsxtrbookindexfirstmark{⟨entry-label⟩}| \quad glossary-bookindex v1.21+ \hfill §8.7.1; 449
\end{quote}

Used by the bookindex style to obtain the first mark and, if found, format it with `\glsxtrbookindexfirstmarkfmt`. 

828
Command Summary

\texttt{\glsxtrbookindexfirstmarkfmt}{(entry-label)} \glsxtrbookindex v1.21+

§8.7.1; 449

Used by the \texttt{bookindex} style to format the first mark.

\texttt{\glsxtrbookindexformatheader}{\langle group-title \rangle} \glsxtrbookindex v1.21+

§8.7.1; 448

Used by the \texttt{bookindex} style to format a group header.

\texttt{\glsxtrbookindexformatsubheader}{\langle previous level \rangle}{\langle level \rangle}{\langle parent-label \rangle}{\langle group-label \rangle}{\langle title \rangle} \glsxtrbookindex v1.49+

Formats the sub-group header.

\texttt{\glsxtrbookindexlastmark}{(entry-label)} \glsxtrbookindex v1.21+

§8.7.1; 449

Used by the \texttt{bookindex} style to obtain the last mark and, if found, format it with \texttt{\glsxtrbookindexlastmarkfmt}.

\texttt{\glsxtrbookindexlastmarkfmt}{(entry-label)} \glsxtrbookindex v1.21+

§8.7.1; 449

Used by the \texttt{bookindex} style to format the last mark.

\texttt{\glsxtrbookindexlocation}{(entry-label)}{\langle location list \rangle} \glsxtrbookindex v1.39+

§8.7.1; 446

Used by the \texttt{bookindex} style to display top-level location lists.

\texttt{\glsxtrbookindexmarkentry}{\langle entry-label \rangle} \glsxtrbookindex v1.21+

§8.7.1; 449

Used by the \texttt{bookindex} style to mark an entry in the aux file.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsxtrbookindexmulticolsenv</td>
<td>Expands to the name of the multicols environment to use.</td>
<td>444</td>
</tr>
<tr>
<td>\glsxtrbookindexname\langle entry-label\rangle</td>
<td>Used by the bookindex style to display a top-level entry’s name.</td>
<td>445</td>
</tr>
<tr>
<td>\glsxtrbookindexparentchildsep</td>
<td>Used by the bookindex style to separate a top-level parent and child entry.</td>
<td>446</td>
</tr>
<tr>
<td>\glsxtrbookindexparentsubchildsep</td>
<td>Used by the bookindex style to separate a sub-level parent and child entry.</td>
<td>446</td>
</tr>
<tr>
<td>\glsxtrbookindexpregroupskip\langle skip\rangle</td>
<td>Used by the bookindex style insert \langle skip\rangle after a group header.</td>
<td>449</td>
</tr>
<tr>
<td>\glsxtrbookindexprelocation\langle entry-label\rangle</td>
<td>Used by the bookindex style to display a separator before top-level location lists.</td>
<td>445</td>
</tr>
<tr>
<td>\glsxtrbookindexsubatendgroup\langle entry-label\rangle</td>
<td>Used by the bookindex style at the end of a letter group (where the last level 1 entry is given by \langle entry-label\rangle).</td>
<td>447</td>
</tr>
<tr>
<td>\glsxtrbookindexsubbetween\langle entry1-label\rangle\langle entry2-label\rangle</td>
<td></td>
<td>447</td>
</tr>
</tbody>
</table>
As \glsxtrbookindexbetween but for level 1 entries.

\glsxtrbookindexsublocation\{⟨entry-label⟩\}\{⟨location list⟩\}
glossary-bookindex v1.39+

§8.7.1; 446

Used by the bookindex style to display child location lists.

\glsxtrbookindexsubname\{⟨entry-label⟩\}
glossary-bookindex v1.21+

§8.7.1; 445

Used by the bookindex style to display a child entry’s name.

\glsxtrbookindexsubprelocation\{⟨entry-label⟩\}
glossary-bookindex v1.21+

§8.7.1; 446

Used by the bookindex style to display a separator before child location lists.

\glsxtrbookindexsubsubatendgroup\{⟨entry-label⟩\}
glossary-bookindex v1.21+

§8.7.1; 447

Used by the bookindex style at the end of a letter group (where the last level 2 entry is given by ⟨entry-label⟩).

\glsxtrbookindexsubsubbetween\{⟨entry1-label⟩\}\{⟨entry2-label⟩\}
glossary-bookindex v1.21+

§8.7.1; 447

As \glsxtrbookindexbetween but for level 2 entries.

\glsxtrcat initial: general

§13; 606

Expands to the default category set by commands like \glsxtr.

\glsxtr\{category\}\{field\}\acs\ supp

(user defined)
Expands to the accessibility support command for the given internal field label and category, which is used by `\glsfieldaccsupp`.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Version</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\glsxtrchecknohyperfirst{⟨entry-label⟩}</code></td>
<td>Sets <code>hyper=false</code> if the <code>nohyperfirst</code> attribute is set.</td>
<td>glossaries-extra v1.07+</td>
<td>§5.1.1; 189</td>
</tr>
<tr>
<td><code>\glsxtrclearlabelprefixes</code></td>
<td>Clears the list of known prefixes.</td>
<td>glossaries-extra-bib2gls v1.37+</td>
<td>§11.5.7; 589</td>
</tr>
<tr>
<td><code>\GlsXtrClearUnsetBuffer</code></td>
<td>Locally clears the buffer, but doesn’t stop buffering.</td>
<td>glossaries-extra v1.49+</td>
<td>§5.10.1; 288</td>
</tr>
<tr>
<td><code>\glsxtrcombiningdiacriticIIrules</code></td>
<td>Expands to the second set of combining diacritic sort rules.</td>
<td>glossaries-extra-bib2gls v1.27+</td>
<td>569</td>
</tr>
<tr>
<td><code>\glsxtrcombiningdiacriticIRules</code></td>
<td>Expands to the first set of combining diacritic sort rules.</td>
<td>glossaries-extra-bib2gls v1.27+</td>
<td>569</td>
</tr>
<tr>
<td><code>\glsxtrcombiningdiacriticIVrules</code></td>
<td>Expands to the fourth set of combining diacritic sort rules.</td>
<td>glossaries-extra-bib2gls v1.27+</td>
<td>569</td>
</tr>
</tbody>
</table>

Expands to the third set of combining diacritic sort rules.
**Command Summary**

\texttt{\textbackslash glsxtrcombiningdiacriticrules} \texttt{glossaries-extra-bib2gls v1.27+}

Expands to all the combining diacritic sort rules.

\texttt{\textbackslash glsxtrcontrolrules} \texttt{glossaries-extra-bib2gls v1.27+}

Expands to control character sort rules.

\texttt{\textbackslash glsxtrcopytoglossary\{\langle entry-label\rangle\}\{\langle glossary-type\rangle\}} \texttt{modifier: *} \texttt{glossaries-extra v1.12+}

Copies the entry to the internal glossary list for the given glossary. The starred version performs a global change. The unstarred version can be localised. Only for use with the “unsrt” family of commands.

\texttt{\textbackslash glsxtr\{counter\}\lofmt\{\langle location\rangle\}\{\langle title\rangle\}} \texttt{(user defined)}

\texttt{\textbackslash glsxtr\{counter\}\lofmt\{\langle location\rangle\}\{\langle title\rangle\}} (user defined)

Used by \texttt{\textbackslash glsxtrdisplayloclnameref} for format a location where the counter matches \langle\textit{counter}\rangle.

\texttt{\textbackslash glsxtrcurrencyrules} \texttt{glossaries-extra-bib2gls v1.27+}

Expands to currency character sort rules.

\texttt{\textbackslash glsxtrcurrentfield} \texttt{glossaries-extra v1.49+}

Placeholder command for use in post-link hooks. This expands to empty if the calling command was one of the \texttt{\textbackslash gls}-like commands or it was one of the inline full form commands, otherwise it will expand to the name of the key associated with the singular form of the command.

\texttt{\textbackslash glsxtrcurrentmglscsname} \texttt{glossaries-extra v1.48+}

\texttt{\textbackslash glsxtrcurrentmglscsname} \texttt{glossaries-extra v1.48+}
Command Summary

Placeholder command for use in multi-entry hooks, this expands to the control sequence name of the calling command.

\glsxtrdefaultentrytextfmt{⟨text⟩}  
\text{glossaries-extra v1.49+}  
§5.5.3; 244

Default inner formatting. Initialised to just do ⟨text⟩.

\GlsXtrDefaultResourceOptions  
\text{glossaries-extra v1.40+}  
§11; 538

Expands to default resource options.

\glsxtrdefaultrevert{⟨text⟩}  
\text{glossaries-extra v1.49+}  
132

The default definition of \glsxtrrevert. Simply does ⟨text⟩.

\GLSxtrdefaultsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}  
\text{glossaries-extra v1.49+}  
174

The default all caps subsequent format style that only shows the short form and insert (with support for \innertextformat).

\Glsxtrdefaultsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}  
\text{glossaries-extra v1.17+}  
174

The default sentence case subsequent format style that only shows the short form and insert (with support for \innertextformat).

\glsxtrdefaultsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}  
\text{glossaries-extra v1.17+}  
174

The default subsequent format style that only shows the short form and insert (with support for \innertextformat).
Command Summary

The default all caps subsequent plural format style that only shows the short form and insert (with support for \innertextformat).

\Glsxtrdefaultsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩} glossaries-extra v1.17+

The default sentence case subsequent plural format style that only shows the short form and insert (with support for \innertextformat).

\glsxtrdefaultsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩} glossaries-extra v1.17+

The default subsequent plural format style that only shows the short form and insert (with support for \innertextformat).

\glsxtrdefffield{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩} glossaries-extra v1.12+

§3.5; 38

Like \GlsXtrSetField but doesn’t perform any existence checks.

\GlsXtrDefineAbbreviationShortcuts

Used by \texttt{shortcuts=abbreviations} and \texttt{shortcuts=all}. This command redefines itself to do nothing because it can only be used once.

\GlsXtrDefineAcronymShortcuts glossaries-extra v1.17+

Used by \texttt{shortcuts=ac} and \texttt{shortcuts=acother}. This command redefines itself to do nothing because it can only be used once.

\GlsXtrDefineOtherShortcuts

Used by \texttt{shortcuts=other} and \texttt{shortcuts=all}. This command redefines itself to do nothing because it can only be used once.
## Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsxtrdetoklocation{(location)}</td>
<td>Just expands to ((location)) by default but may be redefined to help protect awkward characters.</td>
</tr>
<tr>
<td>\glsxtrdigitrules</td>
<td>Expands to 0–9 digit character sort rules (includes superscript and subscript digits).</td>
</tr>
<tr>
<td>\glsxtrdiscardperiod{(entry-label)}{(discarded)}{(no discard)}{(token)}</td>
<td>If ((token)) is a full stop and the entry's category attributes indicate that a full stop should be discarded (such as discardperiod), then ((discarded)) is performed, otherwise ((no discard)) is done and the ((token)) is processed. The actual test to determine if ((token)) is a full stop is performed by \glsxtrifperiod. This command is used in post-link hooks.</td>
</tr>
<tr>
<td>\glsxtrdiscardperiodretainfirstuse{(entry-label)}{(discarded)}{(no discard)}{(token)}</td>
<td>Used to discard a following full stop when the retainfirstuseperiod attribute is set.</td>
</tr>
<tr>
<td>\GlsXtrDiscardUnsetBuffering</td>
<td>Discards the pending buffer and restores \glsunset.</td>
</tr>
<tr>
<td>\glsxtrdisplayendloc{(format)}{(location)}</td>
<td>Used to display an end location from an explicit range.</td>
</tr>
<tr>
<td>\glsxtrdisplayendlochook</td>
<td></td>
</tr>
</tbody>
</table>

---

836
Hook used by \glsxtrdisplayendloc.

\glsxtrdisplaylocnameref{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}{⟨title⟩}{⟨href⟩}{⟨hcounter⟩}{⟨file⟩}

Used to display records created with record=nameref.

\glsxtrdisplaysingleloc{⟨format⟩}{⟨location⟩}

Used to display a single location.

\glsxtrdisplaystartloc{⟨format⟩}{⟨location⟩}

Used to display a start location from an explicit range.

\glsxtrdisplaysupploc{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨src⟩}{⟨location⟩}

modifier: * glossaries-extra-bib2gls v1.36+

Like \glsnoidxdisplayloc but used for supplementary locations.

\glsxtrdoautoindexname{⟨entry-label⟩}{⟨attribute⟩}

Used to automatically index (using \glsxtrautoindex) the entry’s name, if the given attribute is set for the entry’s category.

\glsxtrdopostpunc{⟨code⟩}{⟨token⟩}

If ⟨token⟩ is a recognised punctuation character (see \glsxtrifnextpunc) this does the punctuation character and then ⟨code⟩, otherwise if does ⟨code⟩ followed by ⟨token⟩.

\glsxtrdowrglossaryhook{⟨entry-label⟩}

837
Hook used whenever an entry is indexed. Does nothing by default.

\GlsXtrDualBackLink\{\langle text\rangle\}\{\langle entry-label\rangle\}  
  glossaries-extra-bib2gls v1.30+  \textit{§11.5.7; 587}

Adds a hyperlink to the given entry’s dual (whose label is stored in the field given by \GlsXtrDualField) with the given hyperlink text.

\GlsXtrDualField  \textit{initial: dual}  glossaries-extra-bib2gls v1.30+  \textit{§11.5.7; 588}

Expands to the internal field label used by \GlsXtrDualBackLink.

\glsxtrdeffield\{\langle entry-label\rangle\}\{\langle field-label\rangle\}\{\langle value\rangle\}  glossaries-extra v1.12+  \textit{§3.5; 38}

Like \glsxtrdeffield but (protected) expands \langle value\rangle.

\glsxtrrevert\{\langle text\rangle\}  glossaries-extra v1.49+  158

The definition of \glsxtrrevert used by the emphasized ("em") abbreviation styles. Uses \textup.

\glsxtrsuffix  \textit{initial:} \glsxtrabbrvplural suffix  158

The plural suffix used by the emphasized ("em") abbreviation styles.

\GlsXtrEnableEntryCounting\{\langle category-list\rangle\}\{\langle trigger-value\rangle\}  \textit{§6.1; 311}

Enables entry counting for the given list of categories with the given trigger value (which must be an integer).

\GlsXtrEnableEntryUnitCounting\{\langle category-list\rangle\}\{\langle trigger-value\rangle\}\{\langle counter\rangle\}  \textit{§6.1; 316}
Command Summary

Enables unit entry counting for the given list of categories with the given trigger value (which must be an integer) and the associated counter.

\GlsXtrEnableIndexFormatOverride (preamble only) §12; 602

Allows the format key to override the attribute value.

\GlsXtrEnableInitialTagging\{\langle categories\rangle\}\{\langle cs\rangle\} modifier: * §4.4; 56

Robustly defines the command \langle cs\rangle to accept a single argument, which is a letter (or letters) that needs to be tagged. The unstarred version triggers an error if \langle cs\rangle is already defined. The unstarred version will redefine \langle cs\rangle if it already exists.

\GlsXtrEnableLinkCounting[\langle parent counter\rangle]\{\langle categories\rangle\} glossaries-extra v1.26+ (preamble only) §6.2; 320

Enables link counting for the given categories.

\GlsXtrEnableOnTheFly modifier: * §13; 604

Enables on the fly commands, such as \glsxtr.

\GlsXtrEnablePreLocationTag\{\langle page tag\rangle\}\{\langle pages tag\rangle\} glossaries-extra v1.04+ §8.6.3; 429

Enables the location list tag.

\glsxtrenablerecordcount glossaries-extra v1.21+ §11.4; 552

Redefines the \gls-like commands (except \glsdisp) to use the analogous record count commands (\rgls etc).
\glsxtrfor

When used within \glsxtrforcsvfield signifies that the loop should break at the end of the current iteration.

\glsxtrentryfmt{<entry-label>}{<text>}

As \glsxtrentryfmt but converts \(<text>\) to sentence case.

\glsxtrentryfmt{<entry-label>}{<text>}

Does \(<csname>\){<text>} where the control sequence name \(<csname>\) is obtained from the field given by \GlsXtrFmtField. If hyperref has been loaded and this command will expand to \glsxtrpdfentryfmt\{<entry-label>\}\{<text>\} in a PDF bookmark.

\glsxtrentryparentname{<entry-name>}

Expands to the name field of the given entry’s parent or does nothing if the entry doesn’t have the parent field set or isn’t defined.

\glsxtrequationlofmt{<location>}{<title>}

Used by \glsxtrdisplaylocref to format a location where the counter is equation.

\GlsXtrExpandedFmt{<cs>}{<content>}

Fully-expands \(<content>\) and passes it to \(<cs>\), which must be a command that takes a single argument.

\glsxtrfielddolistloop{<entry-label>}{<field>}

Iterates over the given field’s value using etoolbox’s \dolistcsloop.
Iterates over the given field’s value using `etoolbox`’s `\forlistcsloop`.

 Formats the comma-separated list stored in the given field (identified by its internal label) for the entry identified by `⟨entry-label⟩` using `datatool-base`’s `\DTLformatlist`. This command uses `\glsxtrifhasfield` so the complete list can be obtained with `\glscurrentfield-value`. This adds implicit grouping. There is no starred version.

 Formats the value of the given field, which should be an `etoolbox` internal list, using the same list handler macro as `datatool`’s `\DTLformatlist`.

 Uses `etoolbox`’s `\ifinlistcs` to determine if `⟨item⟩` is in the list stored in the given field.

 Appends `⟨value⟩` to the given field using `etoolbox`’s `\listcsadd`.

 Appends `⟨value⟩` to the given field using `etoolbox`’s `\listcseadd`.
Command Summary

Appends \(\langle value\rangle\) to the given field using etoolbox’s \texttt{\listcsgadd}\.

\[
\texttt{\glsxtrfieldlistxadd}\{\langle entry-label\rangle\}\{\langle field\rangle\}\{\langle value\rangle\}
glossaries-extra v1.12+
\]

§3.5; 39

Appends \(\langle value\rangle\) to the given field using etoolbox’s \texttt{\listcsxadd}\.

\[
\texttt{\glsxtrfieldtitlecase}\{\langle entry-label\rangle\}\{\langle field-label\rangle\}
\]

§5.11; 294

As \texttt{\glsxtrusefield} but converts the field value to title case.

\[
\texttt{\glsxtrfieldtitlecaseds}\{\langle content\rangle\}
glossaries-extra v1.07+
\]

§5.11; 294

Converts \(\langle content\rangle\) to title case (expanding the first token once). Uses \texttt{\glscapitalise}\(\langle content\rangle\)\ if defined, otherwise uses \texttt{\capitalisewords}\. Uses etoolbox’s \texttt{\xifinlistcs} to determine if \(\langle item\rangle\) is in the list stored in the given field.

\[
\texttt{\glsxtrxifinlist}\{\langle entry-label\rangle\}\{\langle field\rangle\}\{\langle item\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}
glossaries-extra v1.12+
\]

§5.14; 308

Maintained for backwards-compatibility used to typeset \(\langle text\rangle\) in small capitals (\texttt{\textsc}\) for the “sc” abbreviation styles on first use.

\[
\texttt{\glsxtrfirstscfont}\{\langle text\rangle\}
glossaries-extra v1.04+
\]

§5.12.2; 301

Maintained for backwards-compatibility used to typeset \(\langle text\rangle\) in a smaller font (\texttt{\texttt{-smaller}}) for the “sm” abbreviation styles on first use.

\[
\texttt{\glsxtrfmt}\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle text\rangle\}
glossaries-extra v1.49+
\]

§5.12.2; 301
As `\glsxtrfmt` but applies a sentence case change to `<text>`.  

\[ \glsxtrfmt[(options)]{(entry-label)}{(text)} \]  

Behaves like `\glslink[(options)]{(entry-label)}{(csname)}{(text)}{(insert)}` where the control sequence name `<csname>` is obtained from the field given by `\GlsXtrFmtField`. The actual format of the link text is governed by `\glsxtrfmtdisplay`.  

\[ \Glsxtrfmt*[[(options)]{(entry-label)}{(text)}{[insert]} \]  

As `\glsxtrfmt*` but applies a sentence case change to `<text>`.  

\[ \glsxtrfmt*[[(options)]{(entry-label)}{(text)}{[insert]} \]  

As the unstarred version `\glsxtrfmt` but accepts the final `<insert>` option.  

\[ \GlsXtrFmtDefaultOptions \]  

`\GlsXtrFmtDefaultOptions` expands to the default options for `\glsxtrfmt`.  

\[ \glsxtrfmtdisplay{(csname)}{(text)}{(insert)} \]  

Formats the link text used in `\glsxtrfmt`.  

\[ \glsxtrfmtexternalnameref{(target)}{(format)}{(title)}{(file)} \]  

Used by `\glsxtrnamereflink` to create an external location hyperlink.  

\[ \GlsXtrFmtField \]  

`\GlsXtrFmtField` expands to the default options for `\glsxtrfmt`.  

\[843\]
## Command Summary

Expands to the name of the used by \glsxtrfmt.

\glsxtrfmtinternalnameref{⟨target⟩}{⟨format⟩}{⟨file⟩}
glossaries-extra-bib2gls v1.37+

§11.5.6; 585

Used by \glsxtrnamereflink to create an internal location hyperlink.

\glsxtrfootnotedesctextname
glossaries-extra v1.42+

144

Expands to the name value for styles like short-footnote-desc.

\glsxtrfootnotedescsort
glossaries-extra v1.42+

145

Expands to the sort value for styles like short-footnote-desc.

\glsxtrfootnotelongformat{⟨entry-label⟩}{⟨fmt-cs⟩}
glossaries-extra v1.49+

145

Used in the footnote text to format the singular long form.

\glsxtrfootnotelongplformat{⟨entry-label⟩}{⟨fmt-cs⟩}
glossaries-extra v1.49+

146

Used in the footnote text to format the plural long form.

\glsxtrfootnotename
glossaries-extra v1.25+

144

Expands to the name value for styles like short-footnote.

\glsxtrforcsvfield{⟨entry-label⟩}{⟨field-label⟩}{⟨handler cs⟩} modifier: *
glossaries-extra v1.24+

§5.13; 304
Command Summary

Iterates over the comma-separated list stored in the given field (identified by its internal label) for the entry identified by \langle entry-label \rangle and performs \langle handler cs \rangle\{\langle element \rangle\} for each element of the list. This command uses \texttt{\glsxtrifhasfield} so the complete list can be obtained with \texttt{\glscurrentfieldvalue}. The unstarred version adds implicit grouping. The starred version doesn’t.

\begin{verbatim}
\begin{verbatim}
\GlsXtrForeignText{\langle entry-label \rangle}{\langle text \rangle}
glossaries-extra v1.32+
\end{verbatim}
\end{verbatim}

If the entry given by \langle entry-label \rangle has the field identified by \texttt{\GlsXtrForeignTextField} then \langle text \rangle will be encapsulated according to the language tag stored in that field (using \texttt{tracklang}’s interface).

\begin{verbatim}
\GlsXtrForeignTextField
\end{verbatim}

Expands to the internal field label used by \texttt{\GlsXtrForeignText}.

\begin{verbatim}
\GlsXtrFormatLocationList{\langle location list \rangle}
\end{verbatim}

Used by \texttt{\glossaryentrynumbers} to encapsulate the entire location list in the glossary.

\begin{verbatim}
\GlsXtrForUnsetBufferedList{\langle handler-cs \rangle}
glossaries-extra v1.31+
\end{verbatim}

Iterates over the labels stored in the current buffer.

\begin{verbatim}
\glsxtrfractionrules
glossaries-extra-bib2gls v1.27+
\end{verbatim}

Expands to the number forms fraction character sort rules.

\begin{verbatim}
\GLSxtrfull[\langle options \rangle]{\langle entry-label \rangle}[\langle insert \rangle]
modifers: * + \langle alt-mod \rangle
\end{verbatim}

As \texttt{\glsxtrfull} but converts the link text to all caps.
As `\glsxtrfull` but converts the first character of the link text to uppercase (for the start of a sentence) using `\makefirstuc`.

References the entry identified by `(entry-label)`. The text produced is obtained from the `short` and `long` values, formatted according to the abbreviation style associated with the entry’s category. The `(insert)` argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. The format produced by this command may not match the format produced by the first use of `\gls{⟨entry-label⟩}`, depending on the abbreviation style. For the first optional argument, see `\glslink` options.

The all caps singular display full form (defined by the abbreviation style).

The sentence case singular display full form (defined by the abbreviation style).

The singular display full form (defined by the abbreviation style).

As `\glsxtrfullpl` but converts the link text to all caps.
As \texttt{\glsxtrfullpl} but converts the first character of the link text to uppercase (for the start of a sentence) using \texttt{\makefirstuc}.

\begin{verbatim}
\glsxtrfullpl\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle insert\rangle\} \textbf{modifiers: } + \langle alt-mod\rangle
\end{verbatim}

References the entry identified by \texttt{\langle entry-label\rangle}. The text produced is obtained from the \texttt{shortplural} and \texttt{longplural} values, formatted according to the abbreviation style associated with the entry’s category. The \texttt{\langle insert\rangle} argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. The format produced by this command may not match the format produced by the first use of \texttt{\glspl\{\langle entry-label\rangle\}}, depending on the abbreviation style. For the first optional argument, see \texttt{\glslink options}.

\begin{verbatim}
\GLSxtrfullplformat\{\langle entry-label\rangle\}\{\langle insert\rangle\} glossaries-extra v1.49+
\end{verbatim}

The all caps plural display full form (defined by the abbreviation style).

\begin{verbatim}
\Glsxtrfullplformat\{\langle entry-label\rangle\}\{\langle insert\rangle\} glossaries-extra v1.49+
\end{verbatim}

The sentence case plural display full form (defined by the abbreviation style).

\begin{verbatim}
\glsxtrfullplformat\{\langle entry-label\rangle\}\{\langle insert\rangle\} glossaries-extra v1.49+
\end{verbatim}

The plural display full form (defined by the abbreviation style).

\begin{verbatim}
\glsxtrfullsaveinsert\{\langle entry-label\rangle\}\{\langle insert\rangle\} glossaries-extra v1.49+
\end{verbatim}

Implemented at the start of all the inline full form commands like \texttt{\glsxtrfull} to save the \texttt{\glsinsert} placeholder. By default, this just does \texttt{\glsxtrssaveinsert}.

\begin{verbatim}
\glsxtrfullsep\{\langle entry-label\rangle\}
\end{verbatim}

Separator used by the parenthetical inline full form and also for some display full forms.
Command Summary

\glsxtrgenabbrvfmt

The display format used by \glsentryfmt for entries that have the short field set and have the regular attribute set to false.

\glsxtrgenentrytextfmt

glossaries-extra v1.49+

Redefined by the \gls-like and \glstext-like hooks to set up the inner formatting. Initialised to \glsxtrdefaultentrytextfmt.

\glsxtrGeneralInitRules
glossaries-extra-bib2gls v1.49+

A shortcut that expands to the ignorable rules, combining diacritic rules, hyphen rules, general punctuation rules, digit rules, and fraction rules.

\glsxtrGeneralLatinAtoGrules
glossaries-extra-bib2gls v1.49+

Expands to the A–G subset of General Latin I sort rules.

\glsxtrGeneralLatinAtoMrules
glossaries-extra-bib2gls v1.49+

Expands to the A–M subset of General Latin I sort rules.

\glsxtrGeneralLatinHtoMrules
glossaries-extra-bib2gls v1.49+

Expands to the H–M subset of General Latin I sort rules.

\glsxtrGeneralLatinIIIRules
glossaries-extra-bib2gls v1.27+

Expands to the third set of General Latin sort rules.
Command Summary

\texttt{\textbackslash glsxtrGeneralLatinIIRules} \quad \texttt{glossaries-extra-bib2gls v1.27+}

Expands to the second set of General Latin sort rules.

\texttt{\textbackslash glsxtrGeneralLatinIRules} \quad \texttt{glossaries-extra-bib2gls v1.27+}

Expands to the first set of General Latin sort rules.

\texttt{\textbackslash glsxtrGeneralLatinIVrules} \quad \texttt{glossaries-extra-bib2gls v1.27+}

Expands to the fourth set of General Latin sort rules.

\texttt{\textbackslash glsxtrGeneralLatinNtoSrules} \quad \texttt{glossaries-extra-bib2gls v1.49+}

Expands to the N–S subset of General Latin I sort rules.

\texttt{\textbackslash glsxtrGeneralLatinNtoZrules} \quad \texttt{glossaries-extra-bib2gls v1.49+}

Expands to the N–Z subset of General Latin I sort rules.

\texttt{\textbackslash glsxtrGeneralLatinTtoZrules} \quad \texttt{glossaries-extra-bib2gls v1.49+}

Expands to the T–Z subset of General Latin I sort rules.

\texttt{\textbackslash glsxtrGeneralLatinVIIIrules} \quad \texttt{glossaries-extra-bib2gls v1.27+}

Expands to the eighth set of General Latin sort rules.

\texttt{\textbackslash glsxtrGeneralLatinVIIRules} \quad \texttt{glossaries-extra-bib2gls v1.27+}

Expands to the seventh set of General Latin sort rules.
Command Summary

\glsxtrgeneralLatinVIRules \ g lexary - extra-bib2gls v1.27+

Expands to the sixth set of General Latin sort rules.

\glsxtrgeneralLatinVIRules \ g lexary - extra-bib2gls v1.27+

Expands to the fifth set of General Latin sort rules.

\glsxtrgeneralpuncaccentsrules \ g lexary - extra-bib2gls v1.49+

Punctuation accent subset of \glsxtrgeneralpuncIrules.

\glsxtrgeneralpuncbracketrules \ g lexary - extra-bib2gls v1.49+

Punctuation bracket subset of \glsxtrgeneralpuncIrules.

\glsxtrgeneralpuncIIrules \ g lexary - extra-bib2gls v1.27+

Expands to the second set of general punctuation (including currency) sort rules.

\glsxtrgeneralpuncIrules \ g lexary - extra-bib2gls v1.27+

Expands to the first set of general punctuation (including currency) sort rules.

\glsxtrgeneralpuncmarksrules \ g lexary - extra-bib2gls v1.49+

Punctuation mark subset of \glsxtrgeneralpuncIrules.

\glsxtrgeneralpuncquoterules \ g lexary - extra-bib2gls v1.49+

Punctuation quote subset of \glsxtrgeneralpuncIrules.

850
Command Summary

\glsxtrgeneralpuncrules{glossaries-extra-bib2gls v1.27+}

Expands to all sets of general punctuation sort rules.

\glsxtrgeneralpuncsignrules{glossaries-extra-bib2gls v1.49+}

Punctuation sign subset of \glsxtrgeneralpuncrules{.}

\glsxtrgetgrouptitle{⟨group-label⟩}{⟨cs⟩}{glossaries-extra v1.14+}

§8.6.4; 432

Obtains the title corresponding to the group identified by ⟨group-label⟩ and stores the result in the control sequence ⟨cs⟩.

\glsxtrgroupfield{initial: group}{glossaries-extra v1.21+}

§8.4.1; 393

Expands to the internal field label used to store the group label (requires record).

\glsxtrglossentry{⟨entry-label⟩}{glossaries-extra v1.21+}

§8.5; 419

Used for standalone entries to display the name with \glossentryname{,} with appropriate hooks.

\glsxtrglossentryother{⟨header⟩}{⟨entry-label⟩}{⟨field-label⟩}

§8.5; 421

Like \glsxtrglossentry{ but uses the given field instead of name.

\GLSxtrheadfirst{⟨entry-label⟩}{glossaries-extra v1.49+}

§5.3.3; 226

Used to display the all caps entry's first field in the page header.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsxtrheadfirst{⟨entry-label⟩}</td>
<td>Used to display the sentence case entry’s first field in the page header (converts to all caps if headuc attribute is true).</td>
<td>§5.3.3; 225</td>
</tr>
<tr>
<td>\glsxtrheadfirst{⟨entry-label⟩}</td>
<td>Used to display the entry’s first field in the page header (converts to all caps if headuc attribute is true).</td>
<td>§5.3.3; 225</td>
</tr>
<tr>
<td>\GLSxtrheadfirstplural{⟨entry-label⟩}</td>
<td>Used to display the all caps entry’s firstplural field in the page header.</td>
<td>§5.3.3; 226</td>
</tr>
<tr>
<td>\Glsxtrheadfirstplural{⟨entry-label⟩}</td>
<td>Used to display the sentence case entry’s firstplural field in the page header (converts to all caps if headuc attribute is true).</td>
<td>§5.3.3; 226</td>
</tr>
<tr>
<td>\GLSxtrheadfull{⟨entry-label⟩}</td>
<td>Used to display the entry’s firstplural field in the page header (converts to all caps if headuc attribute is true).</td>
<td>§5.3.3; 226</td>
</tr>
<tr>
<td>\Glsxtrheadfull{⟨entry-label⟩}</td>
<td>Used to display the entry’s all caps full form in the page header.</td>
<td>§5.3.3; 222</td>
</tr>
<tr>
<td>\Glsxtrheadfull{⟨entry-label⟩}</td>
<td>Used to display the entry’s sentence case full form in the page header (converts to all caps if headuc attribute is true).</td>
<td>§5.3.3; 222</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>\glsxtrheadfull{⟨entry-label⟩}</td>
<td>Used to display the entry’s full form in the page header (converts to all caps if headuc attribute is true).</td>
<td></td>
</tr>
<tr>
<td>\GLSxtrheadfullpl{⟨entry-label⟩}</td>
<td>Used to display the entry’s all caps full plural form in the page header.</td>
<td></td>
</tr>
<tr>
<td>\Glsxtrheadfullpl{⟨entry-label⟩}</td>
<td>Used to display the entry’s sentence case full plural form in the page header (converts to all caps if headuc attribute is true).</td>
<td></td>
</tr>
<tr>
<td>\glsxtrheadfullpl{⟨entry-label⟩}</td>
<td>Used to display the entry’s full plural form in the page header (converts to all caps if headuc attribute is true).</td>
<td></td>
</tr>
<tr>
<td>\GLSxtrheadlong{⟨entry-label⟩}</td>
<td>The behaviour of \GLSfmtlong when it occurs in a page header.</td>
<td></td>
</tr>
<tr>
<td>\Glsxtrheadlong{⟨entry-label⟩}</td>
<td>The behaviour of \Glsfmtlong when it occurs in a page header.</td>
<td></td>
</tr>
<tr>
<td>\glsxtrheadlong{⟨entry-label⟩}</td>
<td>The behaviour of \glsfmtlong when it occurs in a page header.</td>
<td></td>
</tr>
</tbody>
</table>
Command Summary

\GLSxtrheadlongp{\langle entry-label \rangle}
glossaries-extra v1.49+

The behaviour of \GLSfmtlongp when it occurs in a page header.

\Glsxtrheadlongp{\langle entry-label \rangle}
glossaries-extra v1.02+

The behaviour of \Glsfmtlongp when it occurs in a page header.

\glsxtrheadlongp{\langle entry-label \rangle}
glossaries-extra v1.02+

The behaviour of \glsfmtlongp when it occurs in a page header.

\GLSxtrheadnamep{\langle entry-label \rangle}
glossaries-extra v1.49+

Used to display the all caps entry’s name field in the page header.

\Glsxtrheadnamep{\langle entry-label \rangle}
glossaries-extra v1.21+

Used to display the sentence case entry’s name in the page header (converts to all caps if headuc attribute is true).

\glsxtrheadnamep{\langle entry-label \rangle}
glossaries-extra v1.21+

Used to display the entry’s name in the page header (converts to all caps if headuc attribute is true).

\GLSxtrheadpluralp{\langle entry-label \rangle}
glossaries-extra v1.49+

Used to display the all caps entry’s plural field in the page header.
\Glsxtrheadplural\{⟨entry-label⟩\}

Used to display the sentence case entry’s plural field in the page header (converts to all caps if headuc attribute is true).

\glsxtrheadplural\{⟨entry-label⟩\}

Used to display the entry’s plural field in the page header (converts to all caps if headuc attribute is true).

\GLSxtrheadshort\{⟨entry-label⟩\}

The behaviour of \GLSfmtshort when it occurs in a page header.

\Glsxtrheadshort\{⟨entry-label⟩\}

The behaviour of \Glsfmtshort when it occurs in a page header.

\glxsxheadshort\{⟨entry-label⟩\}

The behaviour of \glxsfmtshort when it occurs in a page header.

\GLSxtrheadshortpl\{⟨entry-label⟩\}

The behaviour of \GLSfmtshortpl when it occurs in a page header.

\Glsxtrheadshortpl\{⟨entry-label⟩\}

The behaviour of \Glsfmtshortpl when it occurs in a page header.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsxtrheadshortpl{⟨entry-label⟩}</td>
<td>The behaviour of \glsfmtshortpl when it occurs in a page header.</td>
<td>219</td>
</tr>
<tr>
<td>\GLSxtrheadtext{⟨entry-label⟩}</td>
<td>Used to display the all caps entry’s text field in the page header.</td>
<td>224</td>
</tr>
<tr>
<td>\GLSxtrheadtext{⟨entry-label⟩}</td>
<td>Used to display the sentence case entry’s text field in the page header (converts to all caps if headuc attribute is true).</td>
<td>224</td>
</tr>
<tr>
<td>\glsxtrheadtext{⟨entry-label⟩}</td>
<td>Used to display the entry’s text field in the page header (converts to all caps if headuc attribute is true).</td>
<td>224</td>
</tr>
<tr>
<td>\GLSXTRhiernam{⟨entry-label⟩}</td>
<td>Displays the entry’s hierarchical name where each name is converted to uppercase.</td>
<td>296</td>
</tr>
<tr>
<td>\GLSxthiernam{⟨entry-label⟩}</td>
<td>Displays the entry’s hierarchical name where the first name is converted to uppercase.</td>
<td>296</td>
</tr>
<tr>
<td>\Glsxthiernam{⟨entry-label⟩}</td>
<td>Displays the entry’s hierarchical name where each element name has its first character converted to uppercase.</td>
<td>295</td>
</tr>
</tbody>
</table>
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsxtrhiernamesep</td>
<td>Displays the entry’s hierarchical name using sentence case.</td>
<td>glossaries-extra v1.37+</td>
</tr>
<tr>
<td>\glsxtrhyphenrules</td>
<td>Displays the entry’s hierarchical name.</td>
<td>glossaries-extra bib2gls v1.27+</td>
</tr>
<tr>
<td>\glsxtrhyphensuffixinitial: \glsxtrabbrvpluralsuffix</td>
<td>Separator used by commands like \glsxtrhiernamesep.</td>
<td>glossaries-extra v1.17+</td>
</tr>
<tr>
<td>\glsxtridentifyglslike{⟨label-prefix⟩}{⟨cs⟩}</td>
<td>Expands to hyphen character sort rules.</td>
<td>glossaries-extra v1.37+</td>
</tr>
<tr>
<td>\glsxtriffallcaps{⟨all caps⟩}{⟨not all caps⟩}</td>
<td>The plural suffix used by the “hyphen” abbreviation styles (such as \textit{short-hyphen-long-hyphen}).</td>
<td>glossaries-extra v1.49+</td>
</tr>
<tr>
<td>\glsxtrifcounttrigger{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}</td>
<td>Used to inform bib2gls to include the given command when it searches for dependencies.</td>
<td>glossaries-extra v1.49+</td>
</tr>
</tbody>
</table>
Command Summary

Does \textlangle true\textrangle if the entry’s total use count at the end of the previous run exceeds the trigger value assigned to the entry’s category, otherwise does \textlangle false\textrangle.

\texttt{\textbackslash glsxtricustomdiscardperiod\{\langle true\rangle\}\{\langle false\rangle\}} \quad \textit{initial: \langle false\rangle} \quad \S 5.5.4; 249

User hook to trigger a check for a following full stop. This should do \textlangle true\textrangle if there should be a check for a following full stop otherwise should do \textlangle false\textrangle.

\texttt{\textbackslash glsxtrifemptyglossary\{\langle glossary-type\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}} \quad \S 8; 380

Does \textlangle true\textrangle if the glossary identified by \langle glossary-type\rangle is empty, otherwise does \textlangle false\textrangle. If the glossary doesn’t exist, this does \textlangle true\textrangle and will either generate an error (\texttt{undefaction}=\texttt{error}) or a warning (\texttt{undefaction}=\texttt{warn}). This command considers ignored glossaries as existing.

\texttt{\textbackslash GlsXtrIfFieldCmpNum\{\langle field-label\rangle\}\{\langle entry-label\rangle\}\{\langle op\rangle\}\{\langle number\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}} \quad \textit{modifier: \ast} \quad \S 5.15; 308

Compares the (numeric) value of the field identified by its internal label \langle field-label\rangle for the entry identified by \langle entry-label\rangle with \langle number\rangle where \langle op\rangle is the comparison operator (=, < or >). The unstarred version adds implicit grouping. The starred version doesn’t.

\texttt{\textbackslash GlsXtrIfFieldEqNum\{\langle field-label\rangle\}\{\langle entry-label\rangle\}\{\langle number\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}} \quad \textit{modifier: \ast} \quad \S 5.15; 309

A shortcut that uses \texttt{\textbackslash GlsXtrIfFieldCmpNum} with \langle op\rangle set to =. The unstarred version adds implicit grouping. The starred version doesn’t.

\texttt{\textbackslash GlsXtrIfFieldEqStr\{\langle field-label\rangle\}\{\langle entry-label\rangle\}\{\langle value\rangle\}\{\langle true\rangle\}\{\langle false\rangle\}} \quad \textit{modifier: \ast} \quad \S 5.15; 309

Tests if the entry given by \langle entry-label\rangle has the field identified by its internal label \langle field-label\rangle set to \langle value\rangle. This internally uses \texttt{\textbackslash glsxtrifhasfield} and compares \texttt{\textbackslash glscurrent-}
Command Summary

fieldvalue to ⟨value⟩ using etoolbox’s \ifdefstring. The unstarred version adds implicit grouping. The starred version doesn’t.

\GlsXtrIfFieldEqXpStr{(field-label)\{entry-label\}⟨value⟩\{true\}\{false\}}
 modifier: * glossaries-extra v1.31+

Like \GlsXtrIfFieldEqStr but first (protected) expands ⟨value⟩.

\GlsXtrIfFieldNonZero{(field-label)\{entry-label\}\{true\}\{false\}}
 modifier: * glossaries-extra v1.31+

A shortcut that uses \GlsXtrIfFieldCmpNum to test if the (numeric) value of the field identified by its internal label ⟨field-label⟩ for the entry identified by ⟨entry-label⟩ is non-zero. An empty or undefined field is treated as 0. The unstarred version adds implicit grouping. The starred version doesn’t. The value can be referenced within ⟨true⟩ (where it will be 0) or within ⟨false⟩ using \glscurrentfieldvalue.

\GlsXtrIfFieldUndef{(field-label)\{entry-label\}\{true\}\{false\}}
 glossaries-extra v1.23+

Expandable command that tests if the given field (identified by its internal label) is undefined for the entry given by ⟨entry-label⟩. Internally uses etoolbox’s \ifcsundef command. Unlike \glsxtrifhasfield there is no grouping or starred version.

\GlsXtrIfFieldValueInCsvList{⟨entry-label⟩\{field-label⟩\{csv-list⟩\{true\}\{false\}}
 modifier: * glossaries-extra v1.42+

Tests if the value stored in the given field (identified by its internal label) for the entry identified by ⟨entry-label⟩ is contained in the comma-separated list ⟨csv-list⟩ using \DTLifinlist (provided by datatool-base, which is automatically loaded by the glossaries package). One level expansion is performed on ⟨csv-list⟩. This command uses \glsxtrifhasfield so the field value can be obtained with \glscurrentfieldvalue. The unstarred version adds implicit grouping. The starred version doesn’t.

\glsxtrifhasfield{(field-label)\{entry-label\}\{true\}\{false\}}
 modifier: * glossaries-extra v1.19+
Command Summary

Tests if the field identified by its internal label \(\text{field-label}\) for the entry given by \(\text{entry-label}\) is defined and is not empty. This is like \texttt{\ifglshasfield} but doesn't produce a warning if the entry or field doesn't exist. This sets \texttt{\glsCurrentFieldValue} to the field value and does \(\langle \text{true} \rangle\) if its defined and not empty, otherwise it does \(\langle \text{false} \rangle\). The unstarred version adds implicit grouping to make nesting easier. The starred version doesn't (to make assignments easier).

\[
\texttt{\GlsXtrIfHasNonZeroChildCount}\{\langle \text{entry-label} \rangle\}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\} \quad \text{modifier: } *
\]

Tests if the value in the \texttt{childcount} field is non-zero (using \texttt{\GlsXtrIfFieldNonZero}). This requires the \texttt{save-child-count} resource option.

\[
\texttt{\glsxtrifheaduc}\{\langle \text{entry-label} \rangle\}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\}
\]

If the category associated with the entry given by \(\text{entry-label}\) has the \texttt{headuc} attribute set to \texttt{true} this does \(\langle \text{true} \rangle\) otherwise it does \(\langle \text{false} \rangle\).

\[
\texttt{\glsxtrifhyphenstart}\{\langle \text{text} \rangle\}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\}
\]

If \(\langle \text{text} \rangle\) starts with a hyphen this does \(\langle \text{true} \rangle\) otherwise it does \(\langle \text{false} \rangle\).

\[
\texttt{\glsxtrifindexing}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\}
\]

Tests whether or not the \texttt{noindex} has been set. Does \(\langle \text{false} \rangle\) if \texttt{noindex=true} otherwise does \(\langle \text{true} \rangle\).

\[
\texttt{\GlsXtrIfInGlossary}\{\langle \text{entry-label} \rangle\}\{\langle \text{glossary-type} \rangle\}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\}
\]

Does \(\langle \text{true} \rangle\) if the entry given by \(\langle \text{entry-label} \rangle\) is in the internal list of the glossary identified by \(\langle \text{glossary-type} \rangle\), otherwise it does \(\langle \text{false} \rangle\). If the glossary doesn’t exist, this does \(\langle \text{false} \rangle\) and will either generate an error (\texttt{undefaction=error}) or a warning (\texttt{undefaction=warn}). This command considers ignored glossaries as existing.

860
\glsxtrifinlabelprefixlist{⟨label-prefix⟩}{⟨true⟩}{⟨false⟩}
glossaries-extra-bib2gls v1.37+

Does ⟨true⟩ if ⟨label-prefix⟩ has been identified as a label prefix.

\glsxtrifinmark{⟨true⟩}{⟨false⟩}
glossaries-extra v1.07+

Does ⟨true⟩ if within \markright, \markboth or @starttoc otherwise does ⟨false⟩.

\glsxtrifintoc{⟨true⟩}{⟨false⟩}
glossaries-extra v1.49+

Normally just expands to ⟨false⟩ but @starttoc is redefined to temporarily set this macro to expand to ⟨true⟩ instead. Will always expand to ⟨false⟩ if \glsxtrRevertTocMarks or \glsxtrRevertMarks are used to revert @starttoc to its former definition.

\glsxtrifkeydefined{⟨key⟩}{⟨true⟩}{⟨false⟩}
glossaries-extra v1.12+

Tests if the given ⟨key⟩ has been defined as a glossary entry key. Does ⟨true⟩ if the key has been defined, otherwise does false.

\glsxtriflabelinlist{⟨label⟩}{⟨label-list⟩}{⟨true⟩}{⟨false⟩}
glossaries-extra v1.21+

Does ⟨true⟩ if the given label is in the given comma-separated list of labels, otherwise does ⟨false⟩. The label and list are fully expanded.

\GlsXtrIfLinkCounterDef{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}
glossaries-extra v1.26+

Expands to ⟨true⟩ if the link counter associated with the given entry has been defined, otherwise expands to ⟨false⟩.

\glsxtrifmulti{⟨multi-label⟩}{⟨true⟩}{⟨false⟩}
glossaries-extra v1.48+

Expands to ⟨true⟩ if the link counter associated with the given entry has been defined, otherwise expands to ⟨false⟩.
Command Summary

Does $\langle \text{true} \rangle$ if a multi-entry has been defined with the label $\langle \text{multi-label} \rangle$ otherwise does $\langle \text{false} \rangle$.

\[ \texttt{\glsxtrifnextpunc}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\} \] glossaries-extra §5.5.4; 249

Performs $\langle \text{true} \rangle$ if this command is followed by a recognised punctuation character otherwise does $\langle \text{false} \rangle$. The list of recognised punctuation marks is initialised to . , ; ? ! (full stop, comma, colon, semicolon, question mark, and exclamation mark). Additional punctuation characters can be added with \glsxtraddpunctuationmark.

\[ \texttt{\glsxtrifperiod}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\}\{\langle \text{token} \rangle\} \] glossaries-extra §5.5.4; 249

Does $\langle \text{true} \rangle$ if $\langle \text{token} \rangle$ is a full stop, otherwise does $\langle \text{false} \rangle$.

\[ \texttt{\glsxtrifrecordtrigger}\{\langle \text{entry-label} \rangle\}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\} \] glossaries-extra v1.21+ §11.4; 549

Does $\langle \text{true} \rangle$ if the entry’s total record count (obtained with \glsxtrrecordtriggervalue) exceeds the value supplied by the recordcount attribute, otherwise does $\langle \text{false} \rangle$.

\[ \texttt{\GlsXtrIfUnusedOrUndefined}\{\langle \text{entry-label} \rangle\}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\} \] glossaries-extra v1.34+ §5.10; 286

Does $\langle \text{true} \rangle$ if the entry hasn’t been defined or hasn’t been marked as used, otherwise does $\langle \text{true} \rangle$. Note that this command will generate an error or warning (according to undef-action) if the entry hasn’t been defined, but will still do $\langle \text{true} \rangle$.

\[ \texttt{\GlsXtrIfValueInFieldCsvList}\{\langle \text{entry-label} \rangle\}\{\langle \text{field-label} \rangle\}\{\langle \text{value} \rangle\}\{\langle \text{true} \rangle\}\{\langle \text{false} \rangle\} \] modifier: * glossaries-extra v1.47+ §5.13; 306

Tests if the given value ($\langle \text{value} \rangle$) is contained in the comma-separated list stored in the given field (identified by its internal label) for the entry identified by $\langle \text{entry-label} \rangle$ using \DTLifin-list (provided by datatool-base, which is automatically loaded by the glossaries package). No expansion is performed on $\langle \text{value} \rangle$. This command uses \glsxtrifhasfield so the
Command Summary

Complete list can be obtained with \gls{currentfieldvalue}. The unstarred version adds implicit grouping. The starred version doesn’t.

\glstextrif{wasfirstuse}{\langle true\rangle}{\langle false\rangle}

Initialised by the \glsty{} and \glstext{}-like commands, this expands to \langle true\rangle if the calling command was considered the first use, otherwise it expands to \langle false\rangle. This command may be used within the post-link hook (where it’s too late to test the first use flag with \ifglsused).

\glstextrif{wasglslike}{\langle true\rangle}{\langle false\rangle}
glossaries-extra v1.49+

Initialised by the \glsty{} and \glstext{}-like commands, this expands to \langle true\rangle if the calling command was a \glsty{}-like command, otherwise it expands to \langle false\rangle. This command may be used within the post-link hook.

\glstextrif{wasglslikeandfirstuse}{\langle true\rangle}{\langle false\rangle}
glossaries-extra v1.49+

A shortcut that nests \glstextrif{wasglslike} and \glstextrif{wasfirstuse}. This does \langle true\rangle if the calling command was both a \glsty{}-like command and was considered the first use.

\glstextrif{wassubsequentorshort}{\langle true\rangle}{\langle false\rangle}
glossaries-extra v1.49+

Expands to \langle true\rangle if the calling command was a \glsty{}-like command and was the subsequent use or if \glsxtrcurrentfield was set to short.

\glstextrif{wassubsequentuse}{\langle true\rangle}{\langle false\rangle}
glossaries-extra v1.49+

A shortcut that nests \glstextrif{wasglslike} and \glstextrif{wasfirstuse}. This does \langle true\rangle if the calling command was a \glsty{}-like command but was not considered the first use.

\GlsxtrIfXpFieldEqXpStr{\langle field-label\rangle}{\langle entry-label\rangle}{\langle value\rangle}{\langle true\rangle}{\langle false\rangle}

Modifier: * glossaries-extra v1.31+
Command Summary

Like \GlsXtrIfFieldEqStr but first (protected) expands both the field value and the supplied \langle value \rangle.

\glsxtrIgnorableRules \hspace{1cm} glossaries-extra-bib2gls v1.49+

A shortcut that expands to the control rules, space rules and non-printable rules.

\glsxtrinclinkcounter{\langle counter \rangle} \hspace{1cm} glossaries-extra v1.26+

Increments the link counter with \stepcounter.

\glsxtrindexaliased \hspace{1cm} glossaries-extra v1.12+

Index the current entry’s alias. May only be used within the definition of \glsxtrsetalias-noindex.

\GlsXtrIndexCounterLink{\langle text \rangle}{\langle entry-label \rangle} \hspace{1cm} glossaries-extra-bib2gls v1.29+

Creates a hyperlink (if supported) to the target obtained from indexcounter, if the field has been defined with the given hyperlink text (otherwise just does \langle text \rangle).

\glsxtrindexseealso{\langle entry-label \rangle}{\langle xr-list \rangle} \hspace{1cm} glossaries-extra v1.16+

Indexes the entry identified by \langle entry-label \rangle as a “see also” cross-reference to the entries identified in the comma-separated list \langle xr-list \rangle. The cross-reference list is prefixed with \see-alsoname.

\glsxtrinithyperoutside \hspace{1cm} glossaries-extra v1.21+

Hook that initialises the hyperoutside setting.

§6.2; 320
§5.9.3; 281
§5.1.1; 190
§11.5.8; 599
§5.9.3; 281
§5.1.1; 190
Hook that initialises the \wrgloss setting.

\glsxtrinitwrglossbeforefalse
glossaries-extra v1.14+
Corresponds to \wrgloss=after.

\glsxtrinitwrglossbeforetrue
glossaries-extra v1.14+
Corresponds to \wrgloss=before.

\GLSxtrinlinefullformat{⟨entry-label⟩}{⟨insert⟩}
glossaries-extra v1.49+
Used by \GLSxtrfull to display the all caps inline full form form (defined by the abbreviation style).

\Glsxtrinlinefullformat{⟨entry-label⟩}{⟨insert⟩}
Used by \Glsxtrfull to display the sentence case inline full form form (defined by the abbreviation style).

\glsxtrinlinefullformat{⟨entry-label⟩}{⟨insert⟩}
Used by \glsxtrfull to display the inline full form form (defined by the abbreviation style).

\GLSxtrinlinefullplformat{⟨entry-label⟩}{⟨insert⟩}
glossaries-extra v1.49+
Used by \GLSxtrfullpl to display the plural all caps inline full form form (defined by the abbreviation style).
### Command Summary

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Glsxtrinlinefullplformat {entry-label} {insert}</td>
<td></td>
<td>Used by \Glsxtrfullpl to display the plural sentence case inline full form form (defined by the abbreviation style).</td>
</tr>
<tr>
<td>\glsxtrinlinefullplformat {entry-label} {insert}</td>
<td></td>
<td>Used by \glsxtrfullpl to display the plural inline full form form (defined by the abbreviation style).</td>
</tr>
<tr>
<td>\glsxtrinsertinsidefalse</td>
<td>glossaries-extra v1.02</td>
<td>Sets the \ifglsxtrinsertinside conditional to false.</td>
</tr>
<tr>
<td>\glsxtrinsertinsidetrue</td>
<td>glossaries-extra v1.02</td>
<td>Sets the \ifglsxtrinsertinside to true.</td>
</tr>
<tr>
<td>\GlsXtrInternalLocationHyperlink {counter} {prefix} {location}</td>
<td>glossaries-extra v1.29+</td>
<td>Used by \glsxtrlocationhyperlink to create an internal hyperlink to the given location (advanced command, see documented code for use).</td>
</tr>
<tr>
<td>\glsxtrLatinA</td>
<td>glossaries-extra-bib2gls v1.27+</td>
<td>(Sort rule) expands to the variations of Latin A (includes 0x00AA and 0x2090).</td>
</tr>
<tr>
<td>\glsxtrLatinAA</td>
<td>glossaries-extra-bib2gls v1.27+</td>
<td>(Sort rule) expands to the variations of Latin å.</td>
</tr>
</tbody>
</table>
\texttt{\glsxtrLatinAE}\texttt{Ligature} \hspace{1cm} \texttt{\texttt{glossaries-extra-bib2gls v1.27+}}

(Sort rule) expands to the variations of Latin ae-ligature.

\texttt{\glsxtrLatinE} \hspace{1cm} \texttt{\texttt{glossaries-extra-bib2gls v1.27+}}

(Sort rule) expands to the variations of Latin E (includes 0x2091).

\texttt{\glsxtrLatinEszettSs} \hspace{1cm} \texttt{\texttt{glossaries-extra-bib2gls v1.27+}}

(Sort rule) expands to rule for ß and š.

\texttt{\glsxtrLatinEszettSz} \hspace{1cm} \texttt{\texttt{glossaries-extra-bib2gls v1.27+}}

(Sort rule) expands to rule for ß and ſz.

\texttt{\glsxtrLatinEth} \hspace{1cm} \texttt{\texttt{glossaries-extra-bib2gls v1.27+}}

(Sort rule) expands to the variations of Latin eth.

\texttt{\glsxtrLatinH} \hspace{1cm} \texttt{\texttt{glossaries-extra-bib2gls v1.27+}}

(Sort rule) expands to the variations of Latin H (includes 0x2095).

\texttt{\glsxtrLatinI} \hspace{1cm} \texttt{\texttt{glossaries-extra-bib2gls v1.27+}}

(Sort rule) expands to the variations of Latin I (includes 0x2071).

\texttt{\glsxtrLatinInsularG} \hspace{1cm} \texttt{\texttt{glossaries-extra-bib2gls v1.27+}}

(Sort rule) expands to the variations of Latin insular G and g, G.
\texttt{\textbackslash glsxtrLatinK} \hfill\texttt{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of Latin K (includes 0x2096).

\texttt{\textbackslash glsxtrLatinL} \hfill\texttt{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of Latin L (includes 0x2097).

\texttt{\textbackslash glsxtrLatinLslash} \hfill\texttt{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of Latin ł.

\texttt{\textbackslash glsxtrLatinM} \hfill\texttt{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of Latin M (includes 0x2098).

\texttt{\textbackslash glsxtrLatinN} \hfill\texttt{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of Latin N (includes 0x2099).

\texttt{\textbackslash glsxtrLatinO} \hfill\texttt{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of Latin O (includes 0x00BA and 0x2092).

\texttt{\textbackslash glsxtrLatinOELigature} \hfill\texttt{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of Latin oe-ligature.

\texttt{\textbackslash glsxtrLatinOslash} \hfill\texttt{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of Latin ø.
(Sort rule) expands to the variations of Latin P (includes 0x209A).

(Sort rule) expands to the variations of Latin S (includes 0x209B).

(Sort rule) expands to the variations of Latin schwa.

(Sort rule) expands to the variations of Latin T (includes 0x209C).

(Sort rule) expands to the variations of Latin thorn.

(Sort rule) expands to the variations of Latin wynn.

(Sort rule) expands to the variations of Latin X (includes 0x2093).

Like \GlsXtrSetField but internally uses (etoolbox’s) \cslet instead of \csdef.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{\GlsXtrLetFieldToField} {⟨entry1-label⟩} {⟨field1-label⟩} {⟨entry2-label⟩} {⟨field2-label⟩}</code></td>
<td>Assigns the field identified by its internal label ⟨field1-label⟩ for the entry identified by ⟨entry1-label⟩ to the value of the field identified by ⟨field2-label⟩ for the entry identified by ⟨entry2-label⟩.</td>
</tr>
<tr>
<td><code>{\GlsXtrLinkCounterName} {⟨entry-label⟩}</code></td>
<td>Expands to the name of the link counter associated with the given entry (no check for existence).</td>
</tr>
<tr>
<td><code>{\GlsXtrLinkCounterValue} {⟨entry-label⟩}</code></td>
<td>Expands to the internal link count register associated with the given register or 0 if it hasn’t been defined.</td>
</tr>
<tr>
<td><code>{\GlsXtrLoadResources} [⟨options⟩]</code></td>
<td>A shortcut that uses \glsxtrresourcfile[⟨options⟩]{{basename}}, where the ⟨basename⟩ is obtained from \jobname and \glsxtrresourcetext.</td>
</tr>
<tr>
<td><code>{\glsxtrlocalsetgrouptitle} {⟨group-label⟩} {⟨group-title⟩}</code></td>
<td>Locally assigns the given title ⟨group-title⟩ to the group identified by ⟨group-label⟩.</td>
</tr>
<tr>
<td><code>{\glsxtrlocationanchor}</code></td>
<td>Defined by \glsxtrdisplaylocnameref to expand to the anchor constructed from ⟨counter⟩ and ⟨hcounter⟩, which corresponds to the record counter.</td>
</tr>
<tr>
<td><code>{\GlsXtrLocationField} initial: location</code></td>
<td></td>
</tr>
</tbody>
</table>
Expands to the internal field label used to obtain the formatted location list for the “unsrt” family of commands.

\glsxtrlocationhyperlink{\langle\text{counter}\rangle}{\langle\text{prefix}\rangle}{\langle\text{location}\rangle}
glossaries-extra v1.14+

Used to create a hyperlink to either an external or an internal location, depending on whether or not \glsxtrsupplocationurl is defined and not empty (advanced command, see documented code for use).

\GlsXtrLocationRecordCount{\langle\text{entry-label}\rangle}{\langle\text{counter}\rangle}{\langle\text{location}\rangle}
glossaries-extra v1.21+

Expands to the entry’s record count for the given counter and location (stored in the record-count .\langle\text{counter}\rangle .\langle\text{location}\rangle field) or to 0 if not set.

\glsxtrlocrangefmt
glossaries-extra v1.12+

Expands to the range format (set by \glsxtrdisplaystartloc).

\GLSxtrlong[\langle\text{options}\rangle]{\langle\text{entry-label}\rangle}[\langle\text{insert}\rangle]
modifiers: * + \langle\text{alt-mod}\rangle

As \glsxtrlong but converts the link text to all caps.

\Glsxtrlong[\langle\text{options}\rangle]{\langle\text{entry-label}\rangle}[\langle\text{insert}\rangle]
modifiers: * + \langle\text{alt-mod}\rangle

As \glsxtrlong but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc.

\glsxtrlong[\langle\text{options}\rangle]{\langle\text{entry-label}\rangle}[\langle\text{insert}\rangle]
modifiers: * + \langle\text{alt-mod}\rangle

References the entry identified by \langle\text{entry-label}\rangle. The text produced is obtained from the long value, formatted according to the abbreviation style associated with the entry’s category. The
Command Summary

\(\langle\text{insert}\rangle\) argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \texttt{\glslink} options.

\GLSxtrlongformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle fmt-cs\rangle\} \quad \text{glossaries-extra v1.49+}  

As \texttt{\glsxtrlongformat} but sentence case.

\Glsxtrlongformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle fmt-cs\rangle\} \quad \text{glossaries-extra v1.49+}  

As \texttt{\glsxtrlongformat} but sentence case.

\glsxtrlongformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle fmt-cs\rangle\} \quad \text{glossaries-extra v1.49+}  

Encapsulates the \texttt{long} field for the given entry with \langle fmt-cs\rangle. The \langle insert\rangle argument is the insertion material supplied in the final optional argument of the \texttt{\gls}-like or \texttt{\glstext}-like commands. The \texttt{\ifglsxtrinsertinside}, inner formatting, and accessibility settings are supported.

\GLSxtrlongformatgrp\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle fmt-cs\rangle\} \quad \text{glossaries-extra v1.49+}  

As \texttt{\glsxtrlongformatgrp} but all caps.

\Glsxtrlongformatgrp\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle fmt-cs\rangle\} \quad \text{glossaries-extra v1.49+}  

As \texttt{\glsxtrlongformatgrp} but sentence case.

\glsxtrlongformatgrp\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle fmt-cs\rangle\} \quad \text{glossaries-extra v1.49+}  

As \texttt{\glsxtrlongformatgrp} but adds grouping around \langle insert\rangle (with the inner formatting inside the group).

\glsxtrlonghyphen\{\langle entry-label\rangle\}\{\langle long\rangle\}\{\langle insert\rangle\} \quad \text{glossaries-extra v1.17+}  

872
Command Summary

Formats the long form according to the \texttt{long-hyphen-postshort-hyphen} style.

\texttt{\GLSxtrlonghyphennoshort\{\langle entry-label\rangle\}\{\langle long\rangle\}\{\langle insert\rangle\}}\texttt{ glossaries-extra v1.49+}

As \texttt{\GLSxtrlonghyphennoshort} but converts \texttt{\langle insert\rangle} to all caps. The \texttt{\langle long\rangle} argument should be supplied as all caps.

\texttt{\GLSxtrlonghyphennoshort\{\langle entry-label\rangle\}\{\langle long\rangle\}\{\langle insert\rangle\}}\texttt{ glossaries-extra v1.17+}

Formats the long form according to the \texttt{long-hyphen-noshort-desc-noreg} style.

\texttt{\GLSxtrlonghyphennoshortdescsort}\texttt{ glossaries-extra v1.49+}

Expands to the sort value for the \texttt{long-hyphen-noshort-desc-noreg} styles.

\texttt{\GLSxtrlonghyphennoshortsort}\texttt{ glossaries-extra v1.49+}

Expands to the sort value for the \texttt{long-hyphen-noshort-noreg} styles.

\texttt{\GLSxtrlonghyphenshort\{\langle entry-label\rangle\}\{\langle long\rangle\}\{\langle short\rangle\}\{\langle insert\rangle\}}\texttt{ glossaries-extra v1.49+}

As \texttt{\GLSxtrlonghyphenshort} but converts \texttt{\langle insert\rangle} to all caps. The \texttt{\langle long\rangle} and \texttt{\langle short\rangle} arguments should be supplied as all caps.

\texttt{\GLSxtrlonghyphenshort\{\langle entry-label\rangle\}\{\langle long\rangle\}\{\langle short\rangle\}\{\langle insert\rangle\}}\texttt{ glossaries-extra v1.17+}

Formats the long and short form according to the \texttt{long-hyphen-short-hyphen} style.

\texttt{\GLSxtrlonghyphenshortsort}\texttt{ glossaries-extra v1.49+}
Expands to the sort value for the \textit{long-hyphen-short-hyphen} styles.

\texttt{\textbackslash glsxtrlongnoshortdescname}\hspace{1cm}glossaries-extra v1.25+

Expands to the name value for styles like \textit{long-noshort-desc}.

\texttt{\textbackslash glsxtrlongnoshortname}\hspace{1cm}glossaries-extra v1.25+

Expands to the name value for styles like \textit{long-noshort}.

\texttt{\textbackslash GLSxtrlongpl\langle options\rangle\{\langle entry-label\rangle\}\{\langle insert\rangle\} modifiers: * + \langle alt-mod\rangle}\hspace{1cm}\S 4.3; \textit{53}

As \texttt{\textbackslash glsxtrlongpl} but converts the link text to all caps.

\texttt{\textbackslash Glsxtrlongpl\langle options\rangle\{\langle entry-label\rangle\}\{\langle insert\rangle\} modifiers: * + \langle alt-mod\rangle}\hspace{1cm}\S 4.3; \textit{53}

As \texttt{\textbackslash glsxtrlongpl} but converts the first character of the link text to uppercase (for the start of a sentence) using \texttt{\textbackslash makefirstuc}.

\texttt{\textbackslash glsxtrlongpl\langle options\rangle\{\langle entry-label\rangle\}\{\langle insert\rangle\} modifiers: * + \langle alt-mod\rangle}\hspace{1cm}\S 4.3; \textit{53}

References the entry identified by \texttt{\langle entry-label\rangle}. The text produced is obtained from the \texttt{longplural} value, formatted according to the abbreviation style associated with the entry’s category. The \texttt{\langle insert\rangle} argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \texttt{\textbackslash glslink} options.

\texttt{\textbackslash GLSxtrlongplformat\langle entry-label\rangle\{\langle insert\rangle\}\{\langle fmt-cs\rangle\} glossaries-extra v1.49+}\hspace{1cm}176

As \texttt{\textbackslash glsxtrlongplformat} but all caps.
Command Summary

\Glsxtrlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

As \glsxtrlongformat but sentence case.

\glsxtrlongplformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

As \glsxtrlongplformat but for the \texttt{longplural} field.

\GLSxtrlongplformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

As \glsxtrlongplformatgrp but all caps.

\GLSxtrlongplformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

As \glsxtrlongplformatgrp but sentence case.

\glsxtrlongplformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

As \glsxtrlongplformatgrp but adds grouping around ⟨insert⟩ (with the inner formatting inside the group).

\glsxtrlongshortdescname glossaries-extra v1.17+

Expands to the name value for \texttt{long-short-desc} styles.

\glsxtrlongshortdescsort glossaries-extra v1.04+

Expands to the sort value for \texttt{long-short-desc} styles.

\GLSxtrlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩} glossaries-extra v1.49+
Command Summary

As \glsxtrlongshortformat but all caps.

\Glsxtrlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
glossaries-extra v1.49+

As \glsxtrlongshortformat but sentence case.

\glsxtrlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
glossaries-extra v1.49+

Formats the long form with \glsxtrlongformat and the short form in parentheses with \glsxtrshortformat.

\glsxtrlongshortname
glossaries-extra v1.25+

Expands to the name value for long–short styles.

\GLSxtrlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
glossaries-extra v1.49+

As \glsxtrlongshortplformat but all caps.

\Glsxtrlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
glossaries-extra v1.49+

As \glsxtrlongshortplformat but sentence case.

\glsxtrlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
glossaries-extra v1.49+

As \glsxtrlongshortformat but for the plurals.

876
Command Summary

\glsxtrlongshortscuserdescname \quad glossaries-extra v1.48+

Expands to the value for the \texttt{name} key for styles like \texttt{long-postshort-sc-user-desc}.

\glsxtrlongshortscusername \quad glossaries-extra v1.48+

Expands to the value for the \texttt{name} key for styles like \texttt{long-postshort-sc-user}.

\glsxtrlongshortuserdescname \quad glossaries-extra v1.25+

Expands to the value for the \texttt{name} key for styles like \texttt{long-short-user-desc}.

\glsxtrmarkhook

§5.3.3; 227

Hook that’s performed at the start of \texttt{\markright}, \texttt{\markboth} and \texttt{\@starttoc} to redefine commands that need to change when they occur within page headers or contents. This must be counteracted with \texttt{\glsxtrrestoremarkhook} afterwards.

\glsxtrMathGreekIIrules \quad glossaries-extra-bib2gls v1.27+

Expands to the second set of math Greek sort rules.

\glsxtrMathGreekIrules \quad glossaries-extra-bib2gls v1.27+

Expands to the first set of math Greek sort rules.

\glsxtrMathItalicAlpha \quad glossaries-extra-bib2gls v1.27+

(Sort rule) expands to the variations of math italic Greek alpha.

\glsxtrMathItalicBeta \quad glossaries-extra-bib2gls v1.27+

877
(Sort rule) expands to the variations of math italic Greek beta.

\texttt{\glsxtrMathItalicChi} \quad \text{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of math italic Greek chi.

\texttt{\glsxtrMathItalicDelta} \quad \text{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of math italic Greek delta.

\texttt{\glsxtrMathItalicDigamma} \quad \text{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of math italic Greek digamma.

\texttt{\glsxtrMathItalicEpsilon} \quad \text{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of math italic Greek epsilon.

\texttt{\glsxtrMathItalicEta} \quad \text{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of math italic Greek eta.

\texttt{\glsxtrMathItalicGamma} \quad \text{glossaries-extra-bib2gls v1.27+}

(Sort rule) expands to the variations of math italic Greek gamma.

\texttt{\glsxtrMathItalicGreekIIrules} \quad \text{glossaries-extra-bib2gls v1.27+}

Expands to the second set of math italic Greek sort rules.
Command Summary

\texttt{\glsxtrMathItalicGreekIrules}
glossaries-extra-bib2gls v1.27+

Expands to the first set of math italic Greek sort rules.

\texttt{\glsxtrMathItalicIota}
glossaries-extra-bib2gls v1.27+

(Sort rule) expands to the variations of math italic Greek iota.

\texttt{\glsxtrMathItalicKappa}
glossaries-extra-bib2gls v1.27+

(Sort rule) expands to the variations of math italic Greek kappa.

\texttt{\glsxtrMathItalicLambda}
glossaries-extra-bib2gls v1.27+

(Sort rule) expands to the variations of math italic Greek lambda.

\texttt{\glsxtrMathItalicLowerGreekIIrules}
glossaries-extra-bib2gls v1.27+

Expands to the second set of math italic lowercase Greek sort rules.

\texttt{\glsxtrMathItalicLowerGreekIrules}
glossaries-extra-bib2gls v1.27+

Expands to the first set of math italic lowercase Greek sort rules.

\texttt{\glsxtrMathItalicMu}
glossaries-extra-bib2gls v1.27+

(Sort rule) expands to the variations of math italic Greek mu.

\texttt{\glsxtrMathItalicNabla}
glossaries-extra-bib2gls v1.27+

(Sort rule) expands to the Unicode codepoint for nabla.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\glsxtrMathItalicNu</code></td>
<td>(Sort rule) expands to the variations of math italic Greek nu.</td>
</tr>
<tr>
<td><code>\glsxtrMathItalicOmega</code></td>
<td>(Sort rule) expands to the variations of math italic Greek omega.</td>
</tr>
<tr>
<td><code>\glsxtrMathItalicOmicron</code></td>
<td>(Sort rule) expands to the variations of math italic Greek omicron.</td>
</tr>
<tr>
<td><code>\glsxtrMathItalicPartial</code></td>
<td>(Sort rule) expands to the Unicode codepoint for math italic partial differential.</td>
</tr>
<tr>
<td><code>\glsxtrMathItalicPhi</code></td>
<td>(Sort rule) expands to the variations of math italic Greek phi.</td>
</tr>
<tr>
<td><code>\glsxtrMathItalicPi</code></td>
<td>(Sort rule) expands to the variations of math italic Greek pi.</td>
</tr>
<tr>
<td><code>\glsxtrMathItalicPsi</code></td>
<td>(Sort rule) expands to the variations of math italic Greek psi.</td>
</tr>
<tr>
<td><code>\glsxtrMathItalicRho</code></td>
<td>(Sort rule) expands to the variations of math italic Greek rho.</td>
</tr>
</tbody>
</table>
(Sort rule) expands to the variations of math italic Greek sigma.

(Sort rule) expands to the variations of math italic Greek tau.

(Sort rule) expands to the variations of math italic Greek theta.

Expands to the second set of math italic uppercase Greek sort rules.

Expands to the first set of math italic uppercase Greek sort rules.

(Sort rule) expands to the variations of math italic Greek upsilon.

(Sort rule) expands to the variations of math italic Greek xi.

(Sort rule) expands to the variations of math italic Greek zeta.
**Command Summary**

\texttt{\glsxtrMathUpGreekIIRules} \quad \text{glossaries-extra-bib2gls v1.27+}

Expands to the second set of math upright Greek sort rules.

\texttt{\glsxtrMathUpGreekIIRules} \quad \text{glossaries-extra-bib2gls v1.27+}

Expands to the first set of math upright Greek sort rules.

\texttt{\glsxtrMFUsave} \quad \text{glossaries-extra v1.49+}

Used on the first instance of \texttt{\glsxtrresourcefile}, this will add \texttt{\MFUsave} to the begin document hook and then disable itself. This is provided to help \texttt{bib2gls} pick up any of \texttt{mfiristuc’s exclusions}, blockers and mappings to assist with its sentence case function.

\texttt{\GlsXtrMglsOrGls}\{⟨\texttt{mgls cs}⟩\{⟨\texttt{gls cs}⟩\}⟨\texttt{modifier}⟩\[⟨\texttt{options}⟩\]\{⟨\texttt{label}⟩\}[⟨\texttt{insert}⟩]\}

\text{glossaries-extra v1.48+}

If \langle label \rangle matches a defined multi-entry, this will do \langle mgls cs \rangle otherwise it will do \langle gls cs \rangle. The \langle modifier \rangle (\ast \text{ or } + \text{ or the token identified with } \texttt{\GlsXtrSetAltModifier}) may be omitted.

\texttt{\glsxtrmglsWarnAllSkipped}\{⟨\texttt{message}⟩\{⟨\texttt{insert}⟩\}⟨\texttt{fmt-cs}⟩\}

\text{glossaries-extra v1.48+}

Issues the given warning message with \texttt{\GlossariesExtraWarning} and does \langle fmt-cs⟩\{⟨insert⟩\} (this warning is used if all elements of a multi-entry set are skipped).

\texttt{\GLSxtrmultientryadjustedname}\{⟨\texttt{sublist1}⟩\{⟨\texttt{name}⟩\}⟨\texttt{sublist2}⟩\{⟨\texttt{multi-label}⟩\}

\text{glossaries-extra-bib2gls v1.48+}

As \texttt{\glsxtrmultientryadjustedname} but all caps.
Command Summary

\GlsXtrmultientryadjustedname{(sublist1)}{(name)}{(sublist2)}{(multi-label)}

As \glsxtrmultientryadjustedname but title case.

\Glsxtrmultientryadjustedname{(sublist1)}{(name)}{(sublist2)}{(multi-label)}

As \glsxtrmultientryadjustedname but sentence case.

\glsxtrmultientryadjustednamefmt{(text)}

Used by \compound-adjust-name.

\GLSxtrmultientryadjustednamefmt{(text)}

Used by \GLSxtrmultientryadjustedname to encapsulate the main entry name.

\GlsXtrmultientryadjustednamefmt{(text)}

Used by \GlsXtrmultientryadjustedname to encapsulate the main entry name.

\Glsxtrmultientryadjustednamefmt{(text)}

Used by \Glsxtrmultientryadjustedname to encapsulate the main entry name if the first sublist is empty.

\glsxtrmultientryadjustednamefmt{(text)}

Used by \glsxtrmultientryadjustedname to encapsulate the main entry name.
Command Summary

\GLSxtrmultientryadjustednameother{⟨text⟩}
glossaries-extra-bib2gls v1.48+
§7.14; 377

Used by \GLSxtrmultientryadjustedname to encapsulate the other (not main) entries.

\GlsXtrmultientryadjustednameother{⟨text⟩}
glossaries-extra-bib2gls v1.48+
§7.14; 377

Used by \Glsxtrmultientryadjustedname to encapsulate the other (not main) entries.

\GlSxtrmultientryadjustednameother{⟨text⟩}
glossaries-extra-bib2gls v1.48+
§7.14; 377

Used by \GlSxtrmultientryadjustedname to encapsulate the other (not main) entries.

\glsxtrmultientryadjustednameother{⟨text⟩}
glossaries-extra-bib2gls v1.48+
§7.14; 377

Used by \glsxtrmultientryadjustedname to encapsulate the other (not main) entries.

\glsxtrmultientryadjustednamepostsep{⟨pre-label⟩}{⟨post-label⟩}
glossaries-extra-bib2gls v1.48+
§7.14; 376

Separator used by \glsxtrmultientryadjustedname between the main element and the first element of the second sublist.

\glsxtrmultientryadjustednamepresep{⟨pre-label⟩}{⟨post-label⟩}
glossaries-extra-bib2gls v1.48+
§7.14; 376

Separator used by \glsxtrmultientryadjustedname between the last element of the first sublist and the main element.

\glsxtrmultientryadjustednamesep{⟨pre-label⟩}{⟨post-label⟩}
glossaries-extra-bib2gls v1.48+
§7.14; 376

Separator used by \glsxtrmultientryadjustedname.
Command Summary

\glsxtrmultilastotherindex{⟨multi-label⟩} \hspace{1cm} glossaries-extra v1.48+ §7.13; 374

Expands to the index of the final non-main element in the given multi-entry or nothing if ⟨multi-label⟩ hasn’t been defined.

\glsxtrmultilist{⟨multi-label⟩} \hspace{1cm} glossaries-extra v1.48+ §7.13; 373

Expands to the list of element labels for the multi-entry identified by ⟨multi-label⟩ or nothing if not defined.

\glsxtrmultimain{⟨multi-label⟩} \hspace{1cm} glossaries-extra v1.48+ §7.13; 373

Expands to the main label for the multi-entry identified by ⟨multi-label⟩ or nothing if not defined.

\glsxtrmultimainindex{⟨multi-label⟩} \hspace{1cm} glossaries-extra v1.48+ §7.13; 374

Expands to the index of the main element in the given multi-entry or nothing if ⟨multi-label⟩ hasn’t been defined.

\glsxtrmultisupplocation{(src)}{(location)}{(format)} \hspace{1cm} modifier: * glossaries-extra–bib2gls v1.36+ §11.5.5; 581

Used by \glsxtrdisplaysupploc to format the location.

\glsxtrmultitotalelements{⟨multi-label⟩} \hspace{1cm} glossaries-extra v1.48+ §7.13; 374

Expands to the total number of elements in the given multi-entry or nothing if ⟨multi-label⟩ hasn’t been defined.

\glsxtrnameloclink{(prefix)}{(counter)}{(format)}{(location)}{(text)}{(file)} \hspace{1cm} glossaries-extra–bib2gls v1.37+ §11.5.6; 586

885
Create an external location hyperlink using the prefix and counter.

\glstextrnamereflink\{⟨format⟩\}{⟨title⟩}\{⟨target⟩\}{⟨file⟩}
glossaries-extra-bib2gls v1.37+

Used by \glstextrdisplaylocnameref to create a location hyperlink.

\glstextrnewabbrevpresetkeyhook\{⟨options⟩\}{⟨label⟩}\{⟨short⟩\}
§4.1.5; 45

Hook provided to adjust initialisation within \newabbreviation.

\glstextrnewgls\[⟨default-options⟩\]{⟨prefix⟩}\{⟨cs⟩\}
glossaries-extra v1.21+

Defines the command ⟨cs⟩\[⟨options⟩\]{⟨entry-label⟩} to behave like \gls\[⟨default-options⟩,⟨options⟩\]{⟨prefix⟩⟨entry-label⟩}.

\glstextrnewglsdisp\[⟨default-options⟩\]{⟨prefix⟩}\{⟨cs⟩\}
glossaries-extra v1.49+

Defines the command ⟨cs⟩\[⟨options⟩\]{⟨label⟩}\{⟨text⟩\} to behave like \glsdisp\[⟨default-options⟩,⟨options⟩\]{⟨prefix⟩⟨label⟩}{⟨text⟩}.

\glstextrnewGLSlike\[⟨default-options⟩\]{⟨prefix⟩}{⟨\GLS-like cs⟩}{⟨\GLSpl-like cs⟩}
glossaries-extra v1.21+

Like \glstextrnewgls but provides all caps commands.

\glstextrnewglslike\[⟨default-options⟩\]{⟨prefix⟩}{⟨\gls-like cs⟩}{⟨\glspl-like cs⟩}{⟨\Gls-like cs⟩}{⟨\Glspl-like cs⟩}
glossaries-extra v1.21+

Like \glstextrnewgls but provides plural and sentence case commands as well.
\textbf{Command Summary}

\texttt{\textbackslash glsxtrnewglslink[\langle default-options\rangle]{\langle prefix\rangle}{\langle cs\rangle}} \quad \text{glossaries-extra v1.49+} \quad \S 5.7; 259

Defines the command \texttt{\textbackslash glslink[\langle default-options\rangle]{\langle options\rangle}{\langle label\rangle}{\langle text\rangle}} to behave like \texttt{\textbackslash glsxtrnewglslink[\langle default-options\rangle]{\langle prefix\rangle}{\langle label\rangle}{\langle text\rangle}}.

\texttt{\textbackslash glsxtrnewnumber[\langle key=value list\rangle]{\langle entry-label\rangle}{\langle num\rangle}} \quad \text{(requires \texttt{\usepackage[numbers]{glossaries-extra}})} \quad \S 2.1; 11

Defines a new glossary entry with the given label, \texttt{type} set to \texttt{numbers}, the \texttt{category} set to \texttt{number}, the \texttt{name} set to \texttt{\langle num\rangle} and the \texttt{sort} set to \texttt{\langle entry-label\rangle}. The optional argument is a comma-separated list of glossary entry keys, which can be used to override the defaults.

\texttt{\textbackslash glsxtrnewrgls[\langle default-options\rangle]{\langle prefix\rangle}{\langle cs\rangle}} \quad \text{glossaries-extra v1.21+} \quad \S 5.7; 260

Like \texttt{\textbackslash glsxtrnewgls} but uses \texttt{\textbackslash rgls}.

\texttt{\textbackslash glsxtrnewGLSlike[\langle default-options\rangle]{\langle prefix\rangle}{\langle \texttt{\textbackslash GLS-like} cs\rangle}{\langle \texttt{\textbackslash Glspl-like cs}\rangle}} \quad \text{glossaries-extra v1.21+} \quad \S 5.7; 260

Like \texttt{\textbackslash glsxtrnewrgls} but provides all caps commands.

\texttt{\textbackslash glsxtrnewrglslike[\langle default-options\rangle]{\langle prefix\rangle}{\langle \texttt{\textbackslash g}{\texttt{\textbackslash l}{\texttt{\textbackslash s}{\texttt{\textbackslash p}{\texttt{\textbackslash l}}}}\texttt{\textbackslash i}{\texttt{\textbackslash l}{\texttt{\textbackslash e}{\texttt{\textbackslash t}{\texttt{\textbackslash c}{\texttt{\textbackslash s}}}}}}\texttt{\textbackslash Glspl-like cs}\rangle}{\langle \texttt{\textbackslash Glspl-like cs}\rangle}{\langle \texttt{\textbackslash Glspl-like cs}\rangle}{\langle \texttt{\textbackslash Glspl-like cs}\rangle}{\langle \texttt{\textbackslash Glspl-like cs}\rangle}} \quad \text{glossaries-extra v1.21+} \quad \S 5.7; 260

Like \texttt{\textbackslash glsxtrnewrgls} but provides plural and sentence case commands as well.

\texttt{\textbackslash glsxtrnewsymbol[\langle key=value list\rangle]{\langle entry-label\rangle}{\langle sym\rangle}} \quad \text{(requires \texttt{\usepackage[symbols]{glossaries-extra}})} \quad \S 2.1; 10

Defines a new glossary entry with the given label, \texttt{type} set to \texttt{symbols}, the \texttt{category} set to \texttt{symbol}, the \texttt{name} set to \texttt{\langle sym\rangle} and the \texttt{sort} set to \texttt{\langle entry-label\rangle}. The optional argument is a comma-separated list of glossary entry keys, which can be used to override the defaults.
Command Summary

\texttt{\glsxtrNoGlossaryWarning\{\textit{glossary-type}\}}

Issues a warning with \texttt{\GlossariesExtraWarning} indicating that the given glossary is missing.

\texttt{\GlsXtrNoGlsWarningAutoMake\{\textit{glossary-label}\}}

Advisory message when \texttt{automake} has been used.

\texttt{\GlsXtrNoGlsWarningBuildInfo}

Build advice.

\texttt{\GlsXtrNoGlsWarningCheckFile\{\textit{file}\}}

Advisory message to check the file contents.

\texttt{\GlsXtrNoGlsWarningEmptyMain}

Produces the boilerplate text if the probably empty glossary is the main one.

\texttt{\GlsXtrNoGlsWarningEmptyNotMain\{\textit{glossary-label}\}}

Produces the boilerplate text if the probably empty glossary is not the main one.

\texttt{\GlsXtrNoGlsWarningEmptyStart}

Produces the boilerplate text if a glossary is probably empty.

\texttt{\GlsXtrNoGlsWarningHead\{\textit{glossary-label}\}\{\textit{file}\}}

888
Command Summary

Produces the header boilerplate text if a glossary file is missing.

\texttt{\textbackslash GlsXtrNoGlsWarningMisMatch} \hspace{1cm} \S\,15; 621

Advisory message on mis-matching \texttt{\textbackslash makenoidxglossaries}.

\texttt{\textbackslash GlsXtrNoGlsWarningNoOut\{file\}} \hspace{1cm} \S\,15; 621

Advisory if no output file was created.

\texttt{\textbackslash GlsXtrNoGlsWarningTail} \hspace{1cm} \S\,15; 621

Final paragraph of missing glossary boilerplate text.

\texttt{\textbackslash glsxtrnoidxgroups} \hspace{1cm} \texttt{\texttt{glossaries-extra} v1.49+} \hspace{1cm} \S\,8.4; 389

Makes the group titling mechanism used with the “unsrt” family of commands use the same method as for \texttt{\textbackslash printnoidxglossary} (ASCII only). This command can’t be used with \texttt{\textbackslash makeglossaries} or with \texttt{record}.

\texttt{\textbackslash glsxtrnonprintablerules} \hspace{1cm} \texttt{\texttt{glossaries-extra-bib2gls} v1.27+} \hspace{1cm} 568

Expands to non-printable character sort rules.

\texttt{\textbackslash glsxtrnopostpunc} \hspace{1cm} \texttt{\texttt{glossaries-extra} v1.22+} \hspace{1cm} \S\,8.6.2; 428

When placed at the end of the \texttt{description}, this switches off the post-description punctuation (inserted automatically via options such as \texttt{postdot}) but doesn’t suppress the post-description hook. Does nothing outside of the glossary.

\texttt{\textbackslash glsxtronlydescname} \hspace{1cm} \texttt{\texttt{glossaries-extra} v1.17+} \hspace{1cm} 155
Command Summary

Expands to the name value for styles like long-only-short-only-desc.

\glsxtronlydescsort \textit{glossaries-extra v1.17+}

Expands to the name value for styles like long-only-short-only-desc.

\glsxtronlyname \textit{glossaries v1.25+}

Expands to the name value for styles like long-only-short-only.

\glsxtronlysuffix \textit{initial: glossaries-extra v1.17+}

The plural suffix used by the “only” abbreviation styles (such as long-only-short-only).

\glsxtrOrgKeylist \textit{glossaries-extra v1.49+}

Expands to the original option list as it was supplied to \texttt{newabbreviation}.

\glsxtrOrgShort \textit{glossaries-extra v1.17+}

Expands to the original short form as it was supplied to \texttt{newabbreviation}.

\glsxtrOrgLong \textit{glossaries-extra v1.17+}

Expands to the original long form as it was supplied to \texttt{newabbreviation}.

\GLSxtrp{(field)}{(entry-label)} \textit{glossaries-extra v1.07+}

As \texttt{glsxtrp} but converts to uppercase (but not in the PDF bookmark).
### Command Summary

- **\Glsxtrp{\textless field\textgreater }{\textless entry-label\textgreater }**  
  glossaries-extra v1.07+
  
  As \glsxtrp but converts the first letter to uppercase (but not in the PDF bookmark).

- **\glsxtrp{\textless field\textgreater }{\textless entry-label\textgreater }**  
  glossaries-extra v1.07+
  
  For use in headings and captions (instead of the \gls-like or \glstext-like commands). This command is designed to expand to the field value if used in a PDF bookmark and can also expand to a more appropriate command if it ends up in the page header. Note that there's no optional argument. Options should be set beforehand using \glsxtrsetpopts, which is done automatically in the glossary with \glossxtrsetpopts.

- **\glsxtrpageref{\textless entry-label\textgreater }**  
  glossaries-extra v1.11+
  
  As \glsreffentry but uses \pageref instead of \ref. As with \glsreffentry, this will use \gls instead if the corresponding entry counter is disabled.

- **\glsxtrparen{\textless text\textgreater }**  
  glossaries-extra v1.17+
  
  Used to encapsulate \textless text\textgreater in parentheses.

- **\Glsxtrpdfentryfmt{\textless entry-label\textgreater }{\textless text\textgreater }**  
  glossaries-extra v1.49+
  
  Does \MFUsentencecase{\textless text\textgreater }.

- **\glsxtrpdfentryfmt{\textless entry-label\textgreater }{\textless text\textgreater }**  
  glossaries-extra v1.42+
  
  Just does \textless text\textgreater .

- **\glsxtrpInit{\textless cs-name\textgreater }{\textless entry-label\textgreater }**  
  glossaries-extra v1.51+
  
  §5.4; 231
Hook implemented at the start of \glsxtrp (and case-changing variants) inside the added scoping. By default this disables the post-link hook and ignores its arguments.

\Glsxtrp{⟨gls-options⟩} [⟨dfn-options⟩] {⟨entry-label⟩}  
§13; 605

As \glsxtrp but applies sentence case.

\glsxtrp{⟨gls-options⟩} [⟨dfn-options⟩] {⟨entry-label⟩}  
§13; 605

As \glsxtr but shows the plural form.

\glsxtrpostabbrvfootnote{⟨entry-label⟩} {⟨fmt-code⟩}  
glossaries-extra v1.49+
§4.5.2; 160

Command used in the post-link hook for styles like short-postfootnote.

\glsxtrpostdescabbreviation  
initial: empty
§8.6.2; 428

The default post-description hook for the abbreviation category.

\glsxtrpostdescacronym  
initial: empty
§8.6.2; 427

The default post-description hook for the acronym category.

\glsxtrpostdesc{category}  
glossaries-extra
§8.6.2; 427

The post-description hook associated with the category identified by the label ⟨category⟩.

\glsxtrpostdescgeneral  
initial: empty
§8.6.2; 427

The default post-description hook for the general category.
Command Summary

\glsxtrpostdescindex \usepackage[index]{glossaries-extra} \hspace{1em} initial: empty \hspace{1em} (requires §2.1; 12

The default post-description hook for the index category.

\glsxtrpostdescnumber \usepackage[numbers]{glossaries-extra} \hspace{1em} initial: empty \hspace{1em} (requires §2.1; 11

The default post-description hook for the number category.

\glsxtrpostdescription \hspace{1em} §8.6.2; 427

An additional hook used within \glspostdescription that implements the category post-description hook.

\glsxtrpostdescsymbol \usepackage[symbols]{glossaries-extra} \hspace{1em} initial: empty \hspace{1em} (requires §2.1; 10

The default post-description hook for the symbol category.

\glsxtrpostdescterm \hspace{1em} §8.6.2; 427

The default post-description hook for the term category (which isn’t used by glossaries-extra).

\glsxtrpostfootnotelongformat\{\langle entry-label\rangle\}{\langle fmt-cs\rangle} \hspace{1em} glossaries-extra v1.49+ \hspace{1em} 146

Used in the footnote text to format the long form for styles like short-postfootnote.

\GLSxtrpostphyphenlong\{\langle entry-label\rangle\}{\langle insert\rangle} \hspace{1em} glossaries-extra v1.49+ \hspace{1em} 153

As \glsxtrpostphyphenlong but all caps.
Command Summary

\texttt{\textbackslash glsxtrposthyphenlong\{⟨entry-label⟩\}⟨insert⟩}
glossaries-extra v1.17+

Used within the post-link hook to format the long form according to the \texttt{short-hyphen-postlong-hyphen} style on first use.

\texttt{\textbackslash GLSxtrposthyphenlongpl\{⟨entry-label⟩\}⟨insert⟩}
glossaries-extra v1.49+

As \texttt{\textnormal{\textbackslash glsxtrposthyphenlongpl}} but all caps.

\texttt{\textbackslash glsxtrposthyphenlongpl\{⟨entry-label⟩\}⟨insert⟩}
glossaries-extra v1.49+

As \texttt{\textnormal{\textbackslash glsxtrposthyphenlong}} but shows the plural.

\texttt{\textbackslash GLSxtrposthyphenshort\{⟨entry-label⟩\}⟨insert⟩}
glossaries-extra v1.49+

As \texttt{\textnormal{\textbackslash glsxtrposthyphenshort}} but all caps.

\texttt{\textbackslash glsxtrposthyphenshort\{⟨entry-label⟩\}⟨insert⟩}
glossaries-extra v1.17+

Used within the post-link hook to format the short form according to the \texttt{long-hyphen-postshort-hyphen} style on first use.

\texttt{\textbackslash GLSxtrposthyphenshortpl\{⟨entry-label⟩\}⟨insert⟩}
glossaries-extra v1.49+

As \texttt{\textnormal{\textbackslash glsxtrposthyphenshortpl}} but all caps.

\texttt{\textbackslash glsxtrposthyphenshortpl\{⟨entry-label⟩\}⟨insert⟩}
glossaries-extra v1.49+

As \texttt{\textnormal{\textbackslash glsxtrposthyphenshort}} but plural.
\textit{Command Summary}

\begin{verbatim}
\GLSxtrposthyphensubsequent\{⟨entry-label⟩\}\{⟨insert⟩\} glossaries-extra v1.49+
\end{verbatim}

As \texttt{\GLSxtrposthyphensubsequent} but all caps.

\begin{verbatim}
\GLSxtrposthyphensubsequent\{⟨entry-label⟩\}\{⟨insert⟩\} glossaries-extra v1.17+
\end{verbatim}

Used within the post-link hook to format the insert according to the \texttt{long-hyphen-postshort-hyphen} style on subsequent use.

\begin{verbatim}
\GLSxtrpostlink
\end{verbatim}

A post-link hook that does \texttt{\GLSxtrpostlink\{category\}} if that command has been defined, where the category label is obtained from the entry that has just been referenced with a \texttt{\GLS} or \texttt{\glstext} like command (using \texttt{\GLSlabel}). Does nothing if \texttt{\GLSxtrpostlink\{category\}} isn’t defined.

\begin{verbatim}
\GLSxtrpostlinkAddDescOnFirstUse
\end{verbatim}

May be used within a post-link hook to display the description in parentheses on first use.

\begin{verbatim}
\GLSxtrpostlinkAddSymbolDescOnFirstUse glossaries-extra v1.31+
\end{verbatim}

May be used within a post-link hook to display the symbol and description in parentheses on first use.

\begin{verbatim}
\GLSxtrpostlinkAddSymbolOnFirstUse
\end{verbatim}

May be used within a post-link hook to display the symbol in parentheses on first use.

\begin{verbatim}
\GLSxtrpostlink\{category\}
\end{verbatim}

The post-link hook associated with the category identified by the label \texttt{\{category\}}.
A post-link hook that’s used if a full stop is discarded in order to adjust the space factor (to denote the end of a sentence). If the category post-link hook exists, and will be applied and the full stop will be restored.

A post-link hook that checks if a following full stop needs to be discarded, in which case it does \texttt{\glsxtrpostlinkendsentence}, otherwise it does \texttt{\glsxtrpostlink}.

Used by \texttt{\glsxtrpostlinkAddSymbolDescOnFirstUse} to separate the symbol and description, if both are set.

Hook performed by \texttt{\glsxtrpostlocalreset}. This hook is modified by \texttt{\glsenableentrycount} and \texttt{\glsenableentryunitcount}.

Hook performed by \texttt{\glsxtrpostlocalunset}. This hook is modified by \texttt{\glsenableentrycount} and \texttt{\glsenableentryunitcount}.

Hook added to the end of the \texttt{description} field by the unstarred version of \texttt{\longnewglossaryentry}.

Hook added to the end of the \texttt{category} field by the unstarred version of \texttt{\glsxtrpostname}.
Command Summary

The post-name hook associated with the category identified by the label \langle category\rangle.

\texttt{\glsxtrpostnamehook\{entry-label\}}

§8.6.1; 425

A hook that’s performed within \texttt{\glossentryname} and \texttt{\glossentrynameother} after the entry name is displayed. This hook implements auto-indexing (see §12), then the general hook \texttt{\glsextrapostnamehook} and finally the \texttt{\glsxtrpostname\langle category\rangle} hook.

\texttt{\GlsXtrPostNewAbbreviation}

§4.5.3.1; 165

A hook that’s performed after the entry has been defined.

\texttt{\glsxtrpostreset\{entry-label\}}

§5.10; 283

Hook performed by \texttt{\glsreset}. This hook is modified by \texttt{\glsenableentrycount} and \texttt{\glsenableentryunitcount}.

\texttt{\glsxtrpostunset\{entry-label\}}

§5.10; 283

Hook performed by \texttt{\glsunset}. This hook is modified by \texttt{\glsenableentrycount} and \texttt{\glsenableentryunitcount}.

\texttt{\glsxtrpostuserlongformat\{entry-label\}\{fmt-cs\}}

144
glossaries-extra v1.49+

Formats the long form in parentheses (with \texttt{\glsxtruserparen}) in styles like \texttt{short-postlong-user}.

\texttt{\glsxtrpostusershortformat\{entry-label\}\{fmt-cs\}}

142
glossaries-extra v1.49+

Formats the short form in parentheses (with \texttt{\glsxtruserparen}) in styles like \texttt{long-postshort-user}.
Command Summary

\texttt{\GlsXtrPrefixLabelFallbackLastfalse} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.49+} \hspace{1cm} §11.5.7; 590

Sets the \texttt{\ifGlsXtrPrefixLabelFallbackLast} conditional to false.

\texttt{\GlsXtrPrefixLabelFallbackLasttrue} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.49+} \hspace{1cm} §11.5.7; 590

Sets the \texttt{\ifGlsXtrPrefixLabelFallbackLast} conditional to true.

\texttt{\glsxtrpreglossarystyle} \hspace{1cm} \texttt{glossaries-extra v1.49+} \hspace{1cm} §8.6; 424

Hook performed by \texttt{\setglossarystyle} to initialise default definitions of style commands.

\texttt{\glsxtrprelocation} \hspace{1cm} \texttt{\textit{initial: \space}} \hspace{1cm} \texttt{glossaries-extra-stylemods v1.21+} \hspace{1cm} §8.6.5; 433

Used before the location list in the predefined styles provided by \texttt{glossaries-extra} or modified by \texttt{glossaries-extra-stylemods}.

\texttt{\glsxtrprependlabelprefix\{\langle label-prefix\rangle\}} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.37+} \hspace{1cm} §11.5.7; 589

Prepends \texttt{\langle label-prefix\rangle} to the list of known labels.

\texttt{\GlsXtrPreToDefaultGlsOpts\{\langle options\rangle\}} \hspace{1cm} \texttt{glossaries-extra v1.49+} \hspace{1cm} §5.1.1; 187

Locally prepend \texttt{\langle options\rangle} to the default options for the \texttt{\gls} and \texttt{\glstext}-like commands.

\texttt{\glsxtrprovideaccsuppcmd\{\langle category\rangle\}\{\langle field\rangle\}} \hspace{1cm} \texttt{glossaries-extra v1.42+} \hspace{1cm} (requires \texttt{accsupp})

Defines \texttt{\glsxtr\langle category\rangle\langle field\rangle accsupp} to \texttt{\glsshortaccsupp}, if not already defined.

898
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\GlsXtrProvideBibTeXFields}</td>
<td>Provides the standard \LaTeX fields as glossary entry keys (using \texttt{\glsaddstoragekey}).</td>
</tr>
<tr>
<td>\texttt{\glsxtrprovidecommand{⟨cs⟩}{⟨n⟩}{⟨default⟩}{⟨definition⟩}}</td>
<td>modifier: * Just uses \texttt{\providecommand} within the \LaTeX document but is treated as \texttt{\renewcommand} by \texttt{bib2gls}'s interpreter.</td>
</tr>
<tr>
<td>\texttt{\glsxtrprovidestoragekey{⟨key⟩}{⟨default value⟩}{⟨no link cs⟩}}</td>
<td>modifier: * Like \texttt{\glsaddstoragekey} but does nothing if the key has already been defined.</td>
</tr>
<tr>
<td>\texttt{\glsxtrrecentanchor}</td>
<td>Defined by \texttt{\glsxtrdisplaylocnameref} to expand to the \texttt{⟨href⟩} argument. This corresponds to the value of \texttt{@currentHref} when the record was created.</td>
</tr>
<tr>
<td>\texttt{\GlsXtrRecordCount{⟨entry-label⟩}{⟨counter⟩}}</td>
<td>Expands to the entry’s record count for the given counter (stored in the recordcounter. ⟨counter⟩ field) or to 0 if not set.</td>
</tr>
<tr>
<td>\texttt{\GlsXtrRecordCounter{⟨counter-name⟩}}</td>
<td>(preamble only) Activates recording for the given counter.</td>
</tr>
<tr>
<td>\texttt{\glsxtrrecordtriggervalue{⟨entry-label⟩}}</td>
<td>glossaries-extra v1.21+</td>
</tr>
</tbody>
</table>
Expands to the trigger value used by \glsxtrifrecordtrigger.

\GlsXtrRecordWarning\{⟨glossary-type⟩\} glossaries-extra v1.31+

Incorrect use of \printglossary with non-hybrid record.

\glsxrrregularfont\{⟨text⟩\} glossaries-extra v1.04+ §5.5.2; 241

Used by \glsentryfmt to encapsulate regular entries. Also used by \glsxtrassignfieldfont for regular entries.

\GlsXtrResetLocalBuffer glossaries-extra v1.49+ §5.10.1; 288

If local unset for repeat entries has been enabled with \GlsXtrUnsetBufferEnableRepeatLocal, this will locally reset all entries that are in the buffer that hadn’t been marked as used before the function was enabled.

\glsxtrresourcecount glossaries-extra v1.12+ §11; 536

A count register that is incremented on each use of \GlsXtrLoadResources to provide a unique basename for each resource set.

\glsxtrresourcefile\{⟨options⟩\}\{⟨basename⟩\} glossaries-extra v1.11+ §11; 535

For use with \bib2gls, this both sets up the options for the resource set (which \bib2gls can detect from the aux file) and inputs the file ⟨basename⟩.glstex file created by \bib2gls.

\glsxtrresourceinit glossaries-extra v1.21+ §11; 537

May be defined to temporarily change command definitions before information is written to the aux file by the protected write used by \glsxtrresourcefile.
May be added to the definition of \glsxtrresourceinit to temporarily change the definitions of commands that may be used in regular expressions or within the assign-fields resource option.

\glsxtrrestoremarkhook

Counteracts \glsxtrmarkhook.

\glsxtrrestorepostpunc

Used in the description to counteract the use of \glsxtrnopostpunc. Does nothing outside of the glossary.

\glsxtrrevert\{\langle text\rangle\}

Style-sensitive abbreviation command designed to counteract any font change applied by the style.

\glsxtrRevertMarks

Restores \markright, \markboth and \@starttoc to their previous definitions.

\glsxtrRevertTocMarks

Restores \@starttoc to its previous definition.

\glsxtrsaveinsert\{(entry-label)\}\{\langle insert\rangle\}

Implemented at the start of all the \glstext-like commands (except the inline full form commands like \glsxtrfull) to save the \glsinsert placeholder. By default, this sets \glsinsert to empty.
Command Summary

\glsxtrscfont\{⟨text⟩\}

Maintained for backwards-compatibility used to typeset ⟨text⟩ in small capitals (\textsc) for the “sc” abbreviation styles.

\glsxtrsconlydescname
glossaries-extra v1.48+

Expands to the name value for styles like long-only-short-sc-only-desc.

\glsxtrsconlydescsort
glossaries-extra v1.48+

Expands to the sort value for styles like long-only-short-sc-only-desc.

\glsxtrsconlyname
glossaries-extra v1.48+

Expands to the name value for styles like long-only-short-sc-only.

\glsxtrsc onlyrevert\{(text)\}
glossaries-extra v1.49+

The definition of \glsxtrrevert used by styles like long-only-short-sc-only. Uses \glstextup.

\glsxtrsc onlysuffix  
\textit{initial:} \glsxtrscsuffix
glossaries-extra v1.48+

The plural suffix used by the “sc-only” abbreviation styles (such as long-only-short-sc-only).

\glsxtrrevert\{(text)\}
glossaries-extra v1.49+

The definition of \glsxtrrevert used by the small caps (“sc”) abbreviation styles. Uses \glstextup.
Command Summary

\texttt{\textbackslash glsxtrscsuffix}

The plural suffix used by the small caps (“sc”) abbreviation styles. This switches off the small caps font to prevent the suffix from also appearing in small caps.

\texttt{\textbackslash glsxtrcuserrevert\{⟨text⟩\}}
glossaries-extra v1.49+

The definition of \texttt{\textbackslash glsxtrrevert} used by styles like \texttt{long-postshort-sc-user}. Uses \texttt{\textbackslash glsxtr-screvert}.

\texttt{\textbackslash glsxtrcusersuffix}
glossaries-extra v1.48+

The plural suffix used by styles like \texttt{long-postshort-sc-user}.

\texttt{\textbackslash glsxtrsealsolabels\{⟨entry-label⟩\}}
glossaries-extra v1.16+

\$5.9.2; 280

Expands to the value of the \texttt{seealso} field for the entry identified by \texttt{⟨entry-label⟩}. If the field isn’t set, this will expand to nothing. If the entry isn’t defined, this will expand to \texttt{\relax}.

\texttt{\textbackslash glsxtrseelist\{⟨csv-list⟩\}}
glossaries-extra v1.16+

\$5.13; 302

Fully expands \texttt{⟨csv-list⟩} and passes it to \texttt{\textbackslash glsseelist}.

\texttt{\textbackslash glsxtrseelists\{⟨entry-label⟩\}}
glossaries-extra v1.49+

\$5.9.2; 279

If the entry given by \texttt{⟨entry-label⟩} has the \texttt{see}, \texttt{seealso} or \texttt{alias} fields set, this will display the cross reference according to \texttt{\textbackslash glsxtrseeseeformat} (for \texttt{see} and \texttt{alias}) or \texttt{\textbackslash glsxtrusesesealsoformat} (for \texttt{seealso}). If any of these fields are set, the list is encapsulated with \texttt{\textbackslash glsxtrseelistsencap}.

\texttt{\textbackslash glsxtrseelistsdelim}
glossaries-extra v1.49+

\$5.9.2; 280

903
### Command Summary

**Used by \glsxtrseelists to as separator between sub-lists.**

\[
\glsxtrseelistsencap\langle content\rangle
\]

glossaries-extra v1.49+

**§5.9.2; 279**

**Used by \glsxtrseelists to encapsulate the lists.**

\[
\glsxtrsetactualanchor\langle counter\rangle
\]

glossaries-extra-bib2gls v1.49+

**§11.5.6; 583**

**Hook used by \glsxtrdisplaylocnameref to override the default definition of \glsxtr-actualanchor.**

\[
\GlsXtrSetActualChar\langle character\rangle
\]

(preamble only)

**§12; 603**

**Sets the “actual character” for the auto-indexing feature.**

\[
\glsxtrsetaliasnoindex
\]

glossaries-extra v1.12+

**§5.9.3; 281**

**Hook used to switch off indexing for aliases.**

\[
\GlsXtrSetAltModifier\langle token\rangle\langle options\rangle
\]

**§5; 184**

**Sets \(token\) as a modifier for the \glx-like and \glxttext-like commands that will automatically implement the given options.**

\[
\glsxtrsetbibglssaux\langle basename\rangle
\]

(requires bib2gls v3.0+)

**§2.4; 26**

**As the bibglssaux option.**

\[
\glsxtrsetcategory\langle entry-labels\rangle\langle category-label\rangle
\]

**§10; 517**

904
Globally sets the category field to the fully expanded \textit{(category-label)} for each entry listed in \textit{(entry-labels)}.

\begin{verbatim}
\glsxtrsetcategoryforall{⟨glossary-labels⟩}{⟨category-label⟩}
\end{verbatim}

Globally sets the category field to the fully expanded \textit{(category-label)} for each entry belonging to the glossaries listed in \textit{(glossary-labels)}.

\begin{verbatim}
\glsxtrsetcomplexstyle{⟨entry-label⟩}{⟨n⟩}
glossaries-extra v1.49+
\end{verbatim}

Indicates that the entry given by \textit{(entry-label)} uses a complex abbreviation style. The second argument \textit{(n)} should be numeric, which indicates why it doesn’t work with the variations of \texttt{\glsfirst}: 1 (all caps doesn’t work), 2 (all caps and insert doesn’t work), 3 (insert doesn’t work).

\begin{verbatim}
\GlsXtrSetDefaultGlsOpts{⟨options⟩}
\end{verbatim}

Locally set the default options for the \texttt{\gls}-like and \texttt{\glstext}-like commands.

\begin{verbatim}
\GlsXtrSetDefaultNumberFormat{⟨encap⟩}
glossaries-extra v1.19+
\end{verbatim}

Sets the default \texttt{format} to \textit{(encap)} (without the leading backslash).

\begin{verbatim}
\GlsXtrSetDefaultRangeFormat{⟨encap⟩}
glossaries-extra v1.50+
\end{verbatim}

Sets the default \texttt{format} to \textit{(encap)} (without the leading backslash) for \texttt{\glsstartrange} and \texttt{\glsendrange}.

\begin{verbatim}
\GlsXtrSetEncapChar{⟨character⟩}
\end{verbatim}

(preamble only)

Sets the “encap character” for the auto-indexing feature.
Command Summary

\GlsXtrSetEscChar{⟨character⟩}  (preamble only) §12; 603

Sets the “escape character” for the auto-indexing feature.

\GlsXtrSetField{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}  glossaries-extra v1.12+ §3.5; 39

Assigns ⟨value⟩ to the field identified by its internal label ⟨field-label⟩ for the entry identified by ⟨entry-label⟩. An error (or warning with undefaction=warn) occurs if the entry hasn’t been defined.

\glsxtrsetfieldifexists{⟨entry-label⟩}{⟨field-label⟩}{⟨code⟩}  glossaries-extra v1.12+ §3.5; 39

Used by commands like \GlsXtrSetField to check if the entry exists before assigning a value to the field. The ⟨code⟩ part is the assignment code, which is only done if the required condition is met. This can be redefined if the condition needs to be altered.

\glsxtrsetgrouptitle{⟨group-label⟩}{⟨group-title⟩}  glossaries-extra v1.14+ §8.6.4; 432

Globally assigns the given title ⟨group-title⟩ to the group identified by ⟨group-label⟩.

\glsxtrsetglossarylabel{⟨label⟩}  glossaries-extra v1.39+ §8.3; 382

Sets the label to add (using \label{⟨label⟩}) after the glossary section heading.

\GlsXtrSetLevelChar{⟨character⟩}  (preamble only) §12; 603

Sets the “level character” for the auto-indexing feature.

\glsxtrsetlongfirstuse{⟨entry-label⟩}  glossaries v1.49+ §4.3; 52

Implemented by the \glsxtrlong set of commands to assign \glsxtrifwasfirstuse.
Command Summary

\GlsXtrSetPlusModifier{⟨options⟩}
glossaries-extra v1.49+

Overrides the options that should be implemented by the plus (+) modifier for \gls-like and \glistext-like commands.

\glsxtrsetpopts{⟨options⟩}
glossaries-extra v1.07+

Sets the options that \glsxtrp (and case-change variants) pass to the relevant \glistext-like command.

\glsxtrsetpunctuationmarks{⟨token list⟩}

Sets the punctuation list used by \glsxtrifnextpunc. The ⟨token list⟩ must be a non-delimited list of single tokens that represent each punctuation character. Note that the element of the list must be a single token, which means a single-byte character for pdfL\LaTeX{} (for example, ASCII). Multi-byte characters (UTF-8) will require a native Unicode engine (X\LaTeX{} or LuaL\LaTeX{}).

\GlsXtrSetRecordCountAttribute{⟨category-list⟩}{⟨value⟩}
glossaries-extra v1.21+

Sets the recordcount attribute to ⟨value⟩ for each of the listed categories.

\GlsXtrSetStarModifier{⟨options⟩}
glossaries-extra v1.49+

Overrides the options that should be implemented by the star (*) modifier for \gls-like and \glistext-like commands.

\glsxtrsetupfulldefs

Hook used by \glsxtrfull (and case-changing and plural variations).
\textbf{Command Summary}

\begin{enumerate}
\item \texttt{\glsxtrSetWidest}\{⟨\textit{type}⟩\}{⟨\textit{level}⟩}\glossaries-extra-bib2gls v1.37+ §11.5.3; 579

Written to the \texttt{glstex} by the \texttt{set-widest} option.

\item \texttt{\glsxtrSetWidestFallback}\{⟨\textit{type}⟩\}{⟨\textit{level}⟩}\glossaries-extra-bib2gls v1.37+ §11.5.3; 580

Written to the \texttt{glstex} by the \texttt{set-widest} option if \texttt{bib2gls} can’t determine the widest name.

\item \texttt{\GLSxtrshort}\{⟨\textit{options}⟩\}{⟨\textit{entry-label}⟩}\{⟨\textit{insert}⟩\} modifiers: * + ⟨alt-mod⟩ §4.3; 52

As \texttt{\glsxtrshort} but converts the link text to all caps.

\item \texttt{\Glsxtrshort}\{⟨\textit{options}⟩\}{⟨\textit{entry-label}⟩}\{⟨\textit{insert}⟩\} modifiers: * + ⟨alt-mod⟩ §4.3; 51

As \texttt{\glsxtrshort} but converts the first character of the link text to uppercase (for the start of a sentence) using \texttt{\makefirstuc}.

\item \texttt{\glsxtrshort}\{⟨\textit{options}⟩\}{⟨\textit{entry-label}⟩}\{⟨\textit{insert}⟩\} modifiers: * + ⟨alt-mod⟩ §4.3; 51

References the entry identified by ⟨\textit{entry-label}⟩. The text produced is obtained from the \texttt{short} value, formatted according to the abbreviation style associated with the entry’s category. The ⟨\textit{insert}⟩ argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \texttt{\glslink} options.

\item \texttt{\glsxtrshortdescname}\glossaries-extra v1.17+ 147

Expands to the name value for styles like \texttt{short-nolong-desc}.

\item \texttt{\GLSxtrshortformat}\{⟨\textit{entry-label}⟩\}{⟨\textit{insert}⟩}\{⟨\textit{fmt-cs}⟩\} glossaries-extra v1.49+ 177

As \texttt{\glsxtrshortformat} but all caps.
\end{enumerate}
Command Summary

\Glsxtrshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

As \glsxtrshortformat but sentence case.

\Glsxtrshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

Encapsulates the short field for the given entry with ⟨fmt-cs⟩. The ⟨insert⟩ argument is the insertion material supplied in the final optional argument of the \gls-like or \glstext-like commands. The \texttt{ifglsxtrinsertinside}, inner formatting, and accessibility settings are supported.

\GLSxtrshortformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

As \glsxtrshortformatgrp but all caps.

\GLSxtrshortformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

As \glsxtrshortformatgrp but sentence case.

\GLSxtrshortformatgrp{⟨entry-label⟩}{⟨insert⟩}{⟨fmt-cs⟩} glossaries-extra v1.49+

As \glsxtrshortformatgrp but adds grouping around ⟨insert⟩ (with the inner formatting inside the group).

\glsxtrshorthyphen{⟨short⟩}{⟨entry-label⟩}{⟨insert⟩} glossaries-extra v1.17+

Formats the short form according to the \texttt{short-hyphen-postlong-hyphen} style.

\GLSxtrshorthyphenlong{⟨entry-label⟩}{⟨short⟩}{⟨long⟩}{⟨insert⟩} glossaries-extra v1.49+

As \glsxtrshorthyphenlong but ⟨insert⟩ is converted to all caps.
\glsxtrshorthyphenlong{⟨entry-label⟩}{⟨short⟩}{⟨long⟩}{⟨insert⟩}
glossaries-extra v1.17+

Formats the short and long form according to the short-hyphen-long-hyphen style.

\glsxtrshorthyphenlongsort
glossaries-extra v1.49+

Expands to the sort value for the short-hyphen-long-hyphen styles.

\glsxtrshortlongdescname
glossaries-extra v1.17+

Expands to the name value for short-long-desc styles.

\glsxtrshortlongdescsort
glossaries-extra v1.17+

Expands to the sort value for short-long-desc styles.

\GLSxtrshortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
glossaries-extra v1.49+

As \glsxtrshortlongformat but all caps.

\Glsxtrshortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
glossaries-extra v1.49+

As \glsxtrshortlongformat but sentence case.

\glsxtrshortlongformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}
glossaries-extra v1.49+

Formats the short form with \glsxtrshortformat and the long form in parentheses with \glsxtrlongformat.
**Command Summary**

\texttt{\glsxtrshortlongname} \hspace{1cm} glossaries-extra v1.25+

Expands to the name value for \texttt{short-long} styles.

\texttt{\GLSxtrshortlongplformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle long-fmt-cs\rangle\}\{\langle short-fmt-cs\rangle\}} \hspace{1cm} glossaries-extra v1.49+

As \texttt{\glsxtrshortlongplformat} but all caps.

\texttt{\GLsxtshortlongplformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle long-fmt-cs\rangle\}\{\langle short-fmt-cs\rangle\}} \hspace{1cm} glossaries-extra v1.49+

As \texttt{\glsxtrshortlongplformat} but sentence case.

\texttt{\GLSxtrshortlongplformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle long-fmt-cs\rangle\}\{\langle short-fmt-cs\rangle\}} \hspace{1cm} glossaries-extra v1.49+

As \texttt{\glsxtrshortlongformat} but for the plurals.

\texttt{\glsxtrshortlonguserdescname} \hspace{1cm} glossaries-extra v1.25+

Expands to the value for the \texttt{name} key for styles like \texttt{short-long-user-desc}.

\texttt{\glsxtrshortnolongname} \hspace{1cm} glossaries-extra v1.25+

Expands to the name value for \texttt{short-nolong} styles.

\texttt{\GLSxtrshortpl\{\langle options\rangle\}\{\langle entry-label\rangle\}\{\langle insert\rangle\} \hspace{1cm} \textit{modifiers: \texttt{* + (alt-mod)}}} \hspace{1cm} \S 4.3; 52

As \texttt{\glsxtrshort} but converts the link text to all caps.
As \glsxtrshortpl but converts the first character of the link text to uppercase (for the start of a sentence) using \makefirstuc.

References the entry identified by \langle entry-label \rangle. The text produced is obtained from the \texttt{shortplural} value, formatted according to the abbreviation style associated with the entry’s category. The \langle insert \rangle argument will be inserted at the end of the link text. This command does not alter or depend on the first use flag. For the first optional argument, see \glslink options.

As \glsxtrshortplformat but all caps.

As \glsxtrshortplformat but sentence case.

As \glsxtrshortformat but for the \texttt{shortplural} field.

As \glsxtrshortplformatgrp but all caps.
As \texttt{\textbackslash glsxtrshortplformatgrp} but sentence case.

\texttt{\textbackslash glsxtrshortplformatgrp\{entry-label\}\{insert\}\{fmt-cs\}}
glossaries-extra v1.49+

As \texttt{\textbackslash glsxtrshortplformat} but adds grouping around \texttt{\{insert\}} (with the inner formatting inside the group).

\texttt{\textbackslash glsxtrshowtargetinner\{target-name\}}
glossaries-extra v1.48+

§2.5; 30

Used in inner mode for debugging, this defaults to \texttt{\textbackslash glsshowtargetinner} but is changed by the \texttt{showtargets} options.

\texttt{\textbackslash glsxtrshowtargetouter\{target-name\}}
glossaries-extra v1.48+

§2.5; 30

Used in outer mode for debugging, this defaults to \texttt{\textbackslash glsshowtargetouter} but is changed by the \texttt{showtargets} options.

\texttt{\textbackslash glsxtrshowtargetsymbolleft}
glossaries-extra v1.48+

§2.5; 30

The left marker debugging symbol (\texttt{\texttildetilde}).

\texttt{\textbackslash glsxtrshowtargetsymbolright}
glossaries-extra v1.48+

§2.5; 30

The right marker debugging symbol (\texttt{\textlefttriangle}).

\texttt{\textbackslash glsxtrsmfont\{(text)\}}

Maintained for backwards-compatibility used to typeset \texttt{\{(text)\}} in a smaller font (\texttt{\textbackslash textsmaller}) for the "sm" abbreviation styles.
Command Summary

\glsxtrsmrevert\{\text\}
glossaries-extra v1.49+

The definition of \glsxtrsmrevert used by the smaller (“sm”) abbreviation styles. Uses \text-larger.

\glsxtrsm suffix

The plural suffix used by the smaller (“sm”) abbreviation styles (such as short-sm-long).

\glsxtrspacerules
glossaries-extra-bib2gls v1.27+

Expands to space character sort rules.

\GlsXtrStandaloneEntryHeadName\{\text\}
glossaries-extra v1.49+

§8.5; 423

Used by \glsxtrglossentry for the header and toc.

\GlsXtrStandaloneEntryHeadOther\{\text\}\{\text\}
glossaries-extra v1.49+

§8.5; 423

Used by \glsxtrglossentryother for the header and toc.

\GlsXtrStandaloneEntryName\{\text\}
glossaries-extra v1.37+

§8.5; 420

Used to display the standalone entry name and create the associated hypertarget, if supported.

\GlsXtrStandaloneEntryOther\{\text\}\{\text\}
glossaries-extra v1.37+

§8.5; 421

As \GlsXtrStandaloneEntryName but where the text is obtained from the given field instead of name.
Command Summary

\texttt{\GlsXtrStandaloneEntryPdfName\langle\textit{entry-label}\rangle}\quad \textit{glossaries-extra v1.49+} \hspace{10em} \S8.5; 423

Used by \texttt{\glsxtrglossentry} for the PDF bookmark.

\texttt{\GlsXtrStandaloneEntryPdfOther\langle\textit{entry-label}\rangle\langle\textit{field-label}\rangle}\quad \textit{glossaries-extra v1.49+} \hspace{10em} \S8.5; 423

Used by \texttt{\glsxtrglossentryother} for the PDF bookmark.

\texttt{\GlsXtrStandaloneGlossaryType}\quad \textit{glossaries-extra v1.21+} \hspace{10em} \S8.5; 420

Expands to the glossary type for standalone entries.

\texttt{\GlsXtrStandaloneSubEntryItem\langle\textit{entry-label}\rangle}\quad \textit{glossaries-extra v1.21+} \hspace{10em} \S8.5; 420

Used to display standalone entries that have the \texttt{parent} field set.

\texttt{\glsxtrstarflywarn}\hspace{10em} \S8.5; 423

Issues a warning with \texttt{\GlossariesExtraWarning} indicating that the experimental starred version of \texttt{\GlsXtrEnableOnTheFly} has been used.

\texttt{\GlsXtrStartUnsetBuffering \hspace{10em} \textit{modifier: *}} \texttt{\glossaries-extra v1.30+} \hspace{10em} \S5.10.1; 287

Enables unset buffering. The starred version doesn’t check for duplicates.

\texttt{\GlsXtrStopUnsetBuffering \hspace{10em} \textit{modifier: *}} \texttt{\glossaries-extra v1.30+} \hspace{10em} \S5.10.1; 288

Stops buffering. The starred version performs a global unset.

\texttt{\glsxtrSubScriptDigitrules}\quad \texttt{\glossaries-extra-bib2gls v1.27+} \hspace{10em} \S572

915
Command Summary

Expands to the 0–9 subscript digit character sort rules.

\GLSxtrsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}  
glossaries-extra v1.49+

Used by \glsxtrgenabbrvfmt to display the all caps subsequent singular form (defined by the abbreviation style).

\GLSxtrsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}  
glossaries-extra v1.17+

Used by \glsxtrgenabbrvfmt to display the sentence case subsequent singular form (defined by the abbreviation style).

\glsxtrsubsequentfmt{⟨entry-label⟩}{⟨insert⟩}  
glossaries-extra v1.17+

Used by \glsxtrgenabbrvfmt to display the subsequent singular form (defined by the abbreviation style).

\GLSxtrsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}  
glossaries-extra v1.49+

Used by \glsxtrgenabbrvfmt to display the all caps subsequent plural form (defined by the abbreviation style).

\GLSxtrsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}  
glossaries-extra v1.17+

Used by \glsxtrgenabbrvfmt to display the sentence case subsequent plural form (defined by the abbreviation style).

\glsxtrsubsequentplfmt{⟨entry-label⟩}{⟨insert⟩}  
glossaries-extra v1.17+

Used by \glsxtrgenabbrvfmt to display the subsequent plural form (defined by the abbreviation style).
Command Summary

\texttt{\glsxtrSuperScriptDigitrules} \hfill glossaries-extra-bib2gls v1.27+

Expands to the 0–9 superscript digit character sort rules.

\texttt{\glsxtrsupphypernumber{\langle\text{location}\rangle}} \hfill glossaries-extra v1.14+

Used to hyperlink to a location in an external document if the \texttt{externallocation} attribute has been set. This will define \texttt{\glsxtrsupplocationurl} to the location provided by the attribute or to empty if the attribute isn’t set.

\texttt{\glsxtrsupplocationurl} \hfill glossaries-extra v1.14+

Defined by \texttt{\glsxtrsupphypernumber} to the external location or empty if not provided.

\texttt{\glsxtrtagfont{\langle\text{text}\rangle}} \hfill §4.4; 58

Used by the tagging command defined with \texttt{\GlsXtrEnableInitialTagging}.

\texttt{\glsxtrtaggedlist{\langle\text{singular tag}\rangle}{\langle\text{plural tag}\rangle}{\langle\text{label prefix}\rangle}{\langle\text{csv-list}\rangle}} \hfill §5.13; 303

Similar to \texttt{\glsseelist}, this will start the list with \texttt{\langle\text{singular tag}\rangle} if the list only contains one element and \texttt{\langle\text{plural tag}\rangle} if the list contains more than one element. Each element is prefixed with \texttt{\langle\text{label prefix}\rangle}. The tag is separated from the start of the list with \texttt{\glsxtrtaggedlistsep}. The actual list separators as as for \texttt{\glsseelist}. The \texttt{\langle\text{csv-list}\rangle} is expanded before being iterated over. Does nothing if \texttt{\langle\text{csv-list}\rangle} is empty.

\texttt{\glsxtrtaggedlistsep} \hfill initial: \texttt{\space} glossaries-extra v1.49+

Separator used by \texttt{\glsxtrtaggedlist} between the tag and the list.

\texttt{\glsxtrtarget{\langle\text{entry-label}\rangle}{\langle\text{text}\rangle}} \hfill §5.6; 257

\hfill glossaries-extra v1.51+
Command Summary

Like \glsxtrtarget but only creates the target if the field given by \glsxtrtarget field hasn’t been set (if hyperlinks are supported). If that field hasn’t been set, the target is created and the field is set to the target name.

\glsxtrtarget
\glsxtrtarget field

Expands to the field label used by \glsxtrtarget.

\GlsXtrTheLinkCounter{⟨entry-label⟩}
\GlsXtrTheLinkCounter{⟨entry-label⟩}

Expands to the value of the link counter associated with the given entry or 0 if it hasn’t been defined.

\glsxtrtitlednamereflink{⟨format⟩}{⟨location⟩}{⟨title⟩}{⟨file⟩}
\glsxtrtitlednamereflink{⟨format⟩}{⟨location⟩}{⟨title⟩}{⟨file⟩}

Used by \glsxtrdisplaylocnameref to display locations that have a title and are not associated with the page counter and don’t have an associated \glsxtr⟨counter⟩loctr command. The anchor is obtained from \glsxtrrecentanchor.

\GLSxtrtitlefirst{⟨entry-label⟩}
\GLSxtrtitlefirst{⟨entry-label⟩}

Used to display the all caps entry’s first field in the section title and table of contents.

\Glsxtrtitlefirst{⟨entry-label⟩}
\Glsxtrtitlefirst{⟨entry-label⟩}

Used to display the sentence case entry’s first field in the section title and table of contents.

\glsxtrtitlefirst{⟨entry-label⟩}
\glsxtrtitlefirst{⟨entry-label⟩}

Used to display the entry’s first field in the section title and table of contents.


**Command Summary**

\Glsxtrtitlefirstplural{⟨entry-label⟩}

Used to display the all caps entry’s `firstplural` field in the section title and table of contents.

\Glsxtrtitlefirstplural{⟨entry-label⟩}

Used to display the sentence case entry’s `firstplural` field in the section title and table of contents.

\glsxtrtitlefirstplural{⟨entry-label⟩}

Used to display the entry’s `firstplural` field in the section title and table of contents.

\GLSxtrtitlefull{⟨entry-label⟩}

Used to display the entry’s all caps full form in the section title and table of contents.

\Glsxtrtitlefull{⟨entry-label⟩}

Used to display the entry’s sentence case full form in the section title and table of contents.

\glsxtrtitlefull{⟨entry-label⟩}

Used to display the entry’s full form in the section title and table of contents.

\GLSxtrtitlefullpl{⟨entry-label⟩}

Used to display the entry’s all caps full plural form in the section title and table of contents.
## Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glsxtrtitlefullpl{⟨entry-label⟩}</td>
<td>Used to display the entry’s sentence case full plural form in the section title and table of contents.</td>
</tr>
<tr>
<td>\glsxtrtitlefullpl{⟨entry-label⟩}</td>
<td>Used to display the entry’s full plural form in the section title and table of contents.</td>
</tr>
<tr>
<td>\GLSxtrtitlelong{⟨entry-label⟩}</td>
<td>The normal behaviour of \GLSfmtlong.</td>
</tr>
<tr>
<td>\GLSxtrtitlelong{⟨entry-label⟩}</td>
<td>The normal behaviour of \Glsfmtlong.</td>
</tr>
<tr>
<td>\GLSxtrtitlelongpl{⟨entry-label⟩}</td>
<td>The normal behaviour of \GLSfmtlongpl.</td>
</tr>
<tr>
<td>\GLSxtrtitlelongpl{⟨entry-label⟩}</td>
<td>The normal behaviour of \Glsfmtlongpl.</td>
</tr>
<tr>
<td>\glsxtrtitlelongpl{⟨entry-label⟩}</td>
<td>The normal behaviour of \glsfmtlongpl.</td>
</tr>
</tbody>
</table>
The normal behaviour of `\glsfmtlongpl`.

\begin{verbatim}
\GLSxtrtitlename\{⟨entry-label⟩\}
glossaries-extra v1.42+
\end{verbatim}

Used to display the all caps entry’s name in the section title and table of contents.

\begin{verbatim}
\GLSxtrtitlename\{⟨entry-label⟩\}
glossaries-extra v1.21+
\end{verbatim}

Used to display the sentence case entry’s name in the section title and table of contents.

\begin{verbatim}
\GLSxtrtitlename\{⟨entry-label⟩\}
glossaries-extra v1.21+
\end{verbatim}

Used to display the entry’s name in the section title and table of contents.

\begin{verbatim}
\glsxtrtitleopts
glossaries-extra v1.49+
\end{verbatim}

Expands to the options that commands like `\glsfmtshort` should use in the title or caption within the document text.

\begin{verbatim}
\glsxtrtitleorpdforheading\{⟨title⟩\}{⟨PDF bookmarks⟩}{⟨heading⟩}
glossaries-extra v1.21+
\end{verbatim}

Does the applicable argument depending on whether the command occurs within a title/caption or PDF bookmark or heading.

\begin{verbatim}
\GLSxtrtitleplural\{⟨entry-label⟩\}
glossaries-extra v1.42+
\end{verbatim}

Used to display the all caps entry’s plural field in the section title and table of contents.
Used to display the sentence case entry’s plural field in the section title and table of contents.

\texttt{\glsxtrtitleplural\{\textit{entry-label}\}}

§5.3.3; 224

Used to display the entry’s plural field in the section title and table of contents.

\texttt{\GLSxtrtitleshort\{\textit{entry-label}\}}

\texttt{glossaries-extra v1.42+}

§5.3.3; 218

The normal behaviour of \texttt{\GLSfmtshort}.

\texttt{\Glsxtrtitleshort\{\textit{entry-label}\}}

§5.3.3; 217

The normal behaviour of \texttt{\Glsfmtshort}.

\texttt{\glsxtrtitleshort\{\textit{entry-label}\}}

§5.3.3; 217

The normal behaviour of \texttt{\glsfmtshort}.

\texttt{\GLSxtrtitleshortpl\{\textit{entry-label}\}}

\texttt{glossaries-extra v1.42+}

§5.3.3; 220

The normal behaviour of \texttt{\GLSfmtshortpl}.

\texttt{\Glsxtrtitleshortpl\{\textit{entry-label}\}}

§5.3.3; 219

The normal behaviour of \texttt{\Glsfmtshortpl}.

\texttt{\glsxtrtitleshortpl\{\textit{entry-label}\}}

\texttt{glossaries-extra v1.03+}

§5.3.3; 219

The normal behaviour of \texttt{\glsfmtshortpl}.


**Command Summary**

\texttt{\textbackslash GLSxtrtitletext\{\langle entry-label\rangle\}} \hspace{1cm} glossaries-extra v1.42+

Used to display the all caps entry’s text field in the section title and table of contents.

\texttt{\textbackslash Glsxtrtitletext\{\langle entry-label\rangle\}} \hspace{1cm} §5.3.3; 224

Used to display the sentence case entry’s text field in the section title and table of contents.

\texttt{\textbackslash glsxtrtitletext\{\langle entry-label\rangle\}} \hspace{1cm} §5.3.3; 224

Used to display the entry’s text field in the section title and table of contents.

\texttt{\textbackslash GlSXtrTotalRecordCount\{\langle entry-label\rangle\}} \hspace{1cm} glossaries-extra v1.21+

Expands to the entry’s total record count (stored in the recordcount field) or to 0 if not set.

\texttt{\textbackslash glsxtrtreechildpredesc} \hspace{1cm} initial: \texttt{\textbackslash glstreechildpredesc} \hspace{1cm} glossaries-extra-stylemods v1.46+

Inserted before the child descriptions for the tree styles.

\texttt{\textbackslash glsxtrtreepredesc} \hspace{1cm} initial: \texttt{\textbackslash glstreepredesc} \hspace{1cm} glossaries-extra-stylemods v1.46+

Inserted before the top-level descriptions for the tree styles.

\texttt{\textbackslash glsxtrundefaction\{\langle message\rangle\}\{\langle additional help\rangle\}} \hspace{1cm} glossaries-extra v1.08+

Will either produce an error or a warning, depending on the undefaction setting. In the document environment this will also generate the unknown marker (??).
Command Summary

Expands to the unknown marker (??).

\GlsXtrUnknownDialectWarning\{locale\}\{root language\}  glossaries-extra v1.32+

Issues a warning with \GlossariesExtraWarning indicating that a valid dialect label can’t be determined for the given locale and root language.

\GlsXtrUnsetBufferDisableRepeatLocal  glossaries-extra v1.49+

Disables GlsXtrUnsetBufferEnableRepeatLocal.

\GlsXtrUnsetBufferEnableRepeatLocal  glossaries-extra v1.49+

Allows repeat entries within the buffering code to be locally unset before the link text.

\glsxtrunsrtdo\{entry-label\}  glossaries-extra v1.12+

Used by the “unsrt” family of commands, this displays the glossary entry according to the current glossary style (taking the hierarchical level into account, which may have been adjusted by leveloffset or flatten).

\glsxtrunusedformat\{location\}

The format used by \glsxtraddallcrossrefs.

\glsxtrUpAlpha  glossaries-extra-bib2gls v1.27+

(Sort rule) expands to the variations of math Greek upright alpha.

\glsxtrUpBeta  glossaries-extra-bib2gls v1.27+
(Sort rule) expands to the variations of math Greek upright beta.

\glsxtrUpChi

(Sort rule) expands to the variations of math Greek upright chi.

\glsxtrUpDelta

(Sort rule) expands to the variations of math Greek upright delta.

\glsxtrUpDigamma

(Sort rule) expands to the variations of math Greek upright digamma.

\glsxtrUpEpsilon

(Sort rule) expands to the variations of math Greek upright epsilon.

\glsxtrUpEta

(Sort rule) expands to the variations of math Greek upright eta.

\glsxtrUpGamma

(Sort rule) expands to the variations of math Greek upright gamma.

\glsxtrUpIota

(Sort rule) expands to the variations of math Greek upright iota.
Command Summary

\textit{\textbackslash glsxtUpKappa} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.27+} 
(Sort rule) expands to the variations of math Greek upright kappa.

\textit{\textbackslash glsxtUpLambda} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.27+} 
(Sort rule) expands to the variations of math Greek upright lambda.

\textit{\textbackslash glsxtUpMu} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.27+} 
(Sort rule) expands to the variations of math Greek upright mu.

\textit{\textbackslash glsxtUpNu} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.27+} 
(Sort rule) expands to the variations of math Greek upright nu.

\textit{\textbackslash glsxtUpOmega} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.27+} 
(Sort rule) expands to the variations of math Greek upright omega.

\textit{\textbackslash glsxtUpOmicron} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.27+} 
(Sort rule) expands to the variations of math Greek upright omicron.

\textit{\textbackslash glsxtUpPhi} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.27+} 
(Sort rule) expands to the variations of math Greek upright phi.

\textit{\textbackslash glsxtUpPi} \hspace{1cm} \texttt{glossaries-extra-bib2gls v1.27+} 
(Sort rule) expands to the variations of math Greek upright pi.
\glsxtrUpPsi

(Sort rule) expands to the variations of math Greek upright psi.

\glsxtrUpRho

(Sort rule) expands to the variations of math Greek upright rho.

\glsxtrUpSigma

(Sort rule) expands to the variations of math Greek upright sigma.

\glsxtrUpTau

(Sort rule) expands to the variations of math Greek upright tau.

\glsxtrUpTheta

(Sort rule) expands to the variations of math Greek upright theta.

\glsxtrUpUpsilon

(Sort rule) expands to the variations of math Greek upright upsilon.

\glsxtrUpXi

(Sort rule) expands to the variations of math Greek upright xi.

\glsxtrUpZeta

(Sort rule) expands to the variations of math Greek upright zeta.
\GlsXtrUseAbbrStyleFmts\{⟨style-name⟩\}

Implements the \textit{(display definitions)} code for the given abbreviation style.

\GlsXtrUseAbbrStyleSetup\{⟨style-name⟩\}

Implements the \textit{(setup)} code for the given abbreviation style.

\glsxtrusealias\{⟨entry-label⟩\}

If the entry given by \textit{⟨entry-label⟩} has the \texttt{alias} field set, this will display the cross reference according to \glsxtruseseeformat.

\GLSxtrusefield\{⟨entry-label⟩\}\{⟨field-label⟩\}

As \glsxtrusefield but converts the field value to all caps.

\GLSxtrusefield\{⟨entry-label⟩\}\{⟨field-label⟩\}

As \glsxtrusefield but uses sentence case.

\glsxtrusefield\{⟨entry-label⟩\}\{⟨field-label⟩\}

Expands to the value of the given field (identified by its internal label \textit{⟨field-label⟩}) for the entry given by \textit{⟨entry-label⟩}. Expands to \texttt{\relax} if the entry or field are undefined.

\glsxtruserfield

Expands to the internal label of the field used to store additional information for the “user” abbreviation styles, such as \texttt{long-short-user}. 

928
Command Summary

\texttt{\textbackslash glsxtruserfieldfmt\{\textit{text}\}} \hspace{1cm} \texttt{glossaries-extra v1.49+}

Used to format the value of the field given by \texttt{\textbackslash glsxtruserfield} within \texttt{\textbackslash glsxtruserparen} and \texttt{\textbackslash GLSxtruserparen}.

\texttt{\textbackslash GLSxtruserlongformat\{\textit{entry-label}\}\{\textit{fmt-cs}\}} \hspace{1cm} \texttt{glossaries-extra v1.49+}

As \texttt{\textbackslash glsxtruserlongformat} but all caps.

\texttt{\textbackslash glsxtruserlongformat\{\textit{entry-label}\}\{\textit{fmt-cs}\}} \hspace{1cm} \texttt{glossaries-extra v1.49+}

Formats the singular long form in parentheses (with \texttt{\textbackslash glsxtruserparen}) in styles like short-long-user.

\texttt{\textbackslash GLSxtruserlongplformat\{\textit{entry-label}\}\{\textit{fmt-cs}\}} \hspace{1cm} \texttt{glossaries-extra v1.49+}

As \texttt{\textbackslash glsxtruserlongplformat} but all caps.

\texttt{\textbackslash glsxtruserlongplformat\{\textit{entry-label}\}\{\textit{fmt-cs}\}} \hspace{1cm} \texttt{glossaries-extra v1.49+}

Formats the plural long form in parentheses (with \texttt{\textbackslash glsxtruserparen}) in styles like short-long-user.

\texttt{\textbackslash GLSxtruserlongshortformat\{\textit{entry-label}\}\{\textit{insert}\}\{\textit{long-fmt-cs}\}\{\textit{short-fmt-cs}\}} \hspace{1cm} \texttt{glossaries-extra v1.49+}

As \texttt{\textbackslash glsxtruserlongshortformat} but all caps.

\texttt{\textbackslash glsxtruserlongshortformat\{\textit{entry-label}\}\{\textit{insert}\}\{\textit{long-fmt-cs}\}\{\textit{short-fmt-cs}\}} \hspace{1cm} \texttt{glossaries-extra v1.49+}

As \texttt{\textbackslash glsxtruserlongshortformat} but sentence case.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\glsxtruserlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}</code></td>
<td>Used by styles like <code>long-short-user</code> to format the long and short form.</td>
</tr>
<tr>
<td><code>\GLSxtruserlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}</code></td>
<td>As <code>\glsxtruserlongshortformat</code> but all caps.</td>
</tr>
<tr>
<td><code>\Glsxtruserlongshortplformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}</code></td>
<td>As <code>\glsxtruserlongshortplformat</code> but sentence case.</td>
</tr>
<tr>
<td><code>\glsxtruserlongshortformat{⟨entry-label⟩}{⟨insert⟩}{⟨long-fmt-cs⟩}{⟨short-fmt-cs⟩}</code></td>
<td>Used by styles like <code>long-short-user</code> to format the plural long and plural short form.</td>
</tr>
<tr>
<td><code>\GLSxtruserparen{⟨text⟩}{⟨entry-label⟩}</code></td>
<td>As <code>\glsxtruserparen</code> but the value of the field given by <code>\glsxtruserfield</code> is converted to all caps. The <code>⟨text⟩</code> argument should already be in all caps.</td>
</tr>
<tr>
<td><code>\glsxtruserparen{⟨text⟩}{⟨entry-label⟩}</code></td>
<td>Used by the “user” abbreviation styles, such as <code>long-short-user</code>, to insert the space separator (<code>\glsxtrfullsep</code>) followed by the parenthetical material (<code>\glsxtrparen</code>) consisting of <code>⟨text⟩</code> and, if set, the value of the field given by <code>\glsxtruserfield</code>, separated by <code>\glsxtruserparensep</code>.</td>
</tr>
</tbody>
</table>

Used by styles like `long-short-user` to format the long and short form.

As `\glsxtruserlongshortplformat` but all caps.

As `\glsxtruserlongshortplformat` but sentence case.

Used by styles like `long-short-user` to format the plural long and plural short form.

As `\glsxtruserparen` but the value of the field given by `\glsxtruserfield` is converted to all caps. The `⟨text⟩` argument should already be in all caps.

Used by the “user” abbreviation styles, such as `long-short-user`, to insert the space separator (`\glsxtrfullsep`) followed by the parenthetical material (`\glsxtrparen`) consisting of `⟨text⟩` and, if set, the value of the field given by `\glsxtruserfield`, separated by `\glsxtruserparensep`. 
Command Summary

\glsxtruserparensep \hspace{1cm} \textit{glossaries-extra v1.49+}

The separator used in the parenthetical content of \glsxtruserparen and \GLSxtruserparen.

\GLSxtrusershortformat\{\langle entry-label\rangle\}\{\langle fmt-cs\rangle\} \hspace{1cm} \textit{glossaries-extra v1.49+}

As \glsxtrusershortformat but all caps.

\glsxtrusershortformat\{\langle entry-label\rangle\}\{\langle fmt-cs\rangle\} \hspace{1cm} \textit{glossaries-extra v1.49+}

Formats the singular short form in parentheses (with \glsxtruserparen) in styles like \textit{long-short-user}.

\GLSxtrusershortlongformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle long-fmt-cs\rangle\}\{\langle short-fmt-cs\rangle\} \hspace{1cm} \textit{glossaries-extra v1.49+}

As \glsxtrusershortlongformat but all caps.

\Glsxtrusershortlongformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle long-fmt-cs\rangle\}\{\langle short-fmt-cs\rangle\} \hspace{1cm} \textit{glossaries-extra v1.49+}

As \glsxtrusershortlongformat but sentence case.

\glsxtrusershortlongformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle long-fmt-cs\rangle\}\{\langle short-fmt-cs\rangle\} \hspace{1cm} \textit{glossaries-extra v1.49+}

Used by styles like \textit{short-long-user} to format the short and long form.

\GLSxtrusershortlongplformat\{\langle entry-label\rangle\}\{\langle insert\rangle\}\{\langle long-fmt-cs\rangle\}\{\langle short-fmt-cs\rangle\} \hspace{1cm} \textit{glossaries-extra v1.49+}

931
Command Summary

As \texttt{\textbackslash glsxtrusersongplformat} but all caps.

\begin{verbatim}
\texttt{\textbackslash Glxtrusersongplformat\{entry-label\}\{insert\}\{long-fmt-cs\}\{short-fmt-cs\}}
glossaries-extra v1.49+
\end{verbatim}

As \texttt{\textbackslash glsxtrusersongplformat} but sentence case.

\begin{verbatim}
\texttt{\textbackslash Glxtrusersongplformat\{entry-label\}\{insert\}\{long-fmt-cs\}\{short-fmt-cs\}}
glossaries-extra v1.49+
\end{verbatim}

Used by styles like \texttt{short-long-user} to format the plural short and plural long form.

\begin{verbatim}
\texttt{\textbackslash GLSxtrusersongplformat\{entry-label\}\{fmt-cs\}}
glossaries-extra v1.49+
\end{verbatim}

As \texttt{\textbackslash glsxtrusersongplformat} but all caps.

\begin{verbatim}
\texttt{\textbackslash Glxtrusersongplformat\{entry-label\}\{fmt-cs\}}
glossaries-extra v1.49+
\end{verbatim}

Formats the plural short form in parentheses (with \texttt{\textbackslash glsxtruserparen}) in styles like \texttt{long-short-user}.

\begin{verbatim}
\texttt{\textbackslash glsxusersuffix initial: \textbackslash glsxtrabvpluralsuffix}
glossaries-extra v1.04+
\end{verbatim}

The plural suffix used by styles like \texttt{short-long-user}.

\begin{verbatim}
\texttt{\textbackslash glsxusesee\{entry-label\}}
glossaries-extra v1.06+
\end{verbatim}

If the entry given by \texttt{\{entry-label\}} has the \texttt{see} field set, this will display the cross reference according to \texttt{\textbackslash glsxuseseeformat}.

\begin{verbatim}
\texttt{\textbackslash glsxuseseesalso\{entry-label\}}
glossaries-extra v1.16+
\end{verbatim}

§5.9.2; 280

§5.9.2; 280

932
If the entry given by \entrylabel has the seealso field set, this will display the cross-reference according to \glsxtruseealsoformat.

\glsxtruseealsoformat{\csvlist} \textit{glossaries-extra v1.16+} \S5.9.2; 281

Formats the comma-separated list of entry labels as a “see also” cross-reference.

\glsxtruseeformat{\tag}{\xrlist} \textit{glossaries-extra v1.06+}

Format used by \glsxtrusee. This internally uses \glsseeformat.

\GlslXtrWarn Deprecated Abbr\textit{Style}{\oldname}{\newname} \textit{glossaries-extra v1.04+} \S4.5.2; 160

Issues a warning with \GlossariesExtraWarning indicating that a deprecated abbreviation style has been used.

\GlslXtrWarning{\options}{\entry} \textit{GlossariesExtraWarning} \S13; 605

Issues a warning with \GlossariesExtraWarning indicating that the given options list has been ignored by the given entry because it has already been defined.

\glsxtrword{\word} \textit{glossaries-extra v1.17+} \S10.2.1.1; 521

Used to mark each word by the markwords and markshortwords attributes.

\glsxtrwordsep \textit{glossaries-extra v1.17+} \S10.2.1.1; 521

Used to mark word separator space by the markwords and markshortwords attributes.

\glsxtrwordsephyphen \textit{glossaries-extra v1.49+}
Command Summary

Used to mark compound word separator hyphen by the `markwords` and `markshortwords` attributes.

\[\texttt{\glsxtrwrglossaryhook\{}\langle\text{entry-label}\rangle\}\]  
Hook implemented everytime an entry is indexed.

\[\texttt{\glsxtrwrglossarylocfmt\{}\langle\text{location}\rangle\{\langle\text{title}\rangle\}\}\]  
Used by `\glsxtrdisplaylocnameref` to format a location where the counter is wrglossary.

\[\texttt{\glsxtrwrglosscountermark\{}\langle\text{number}\rangle\}\]  
Used to mark where the wrglossary counter is incremented with `debug=showwrgloss`.

\[\texttt{\glsxtrwrglossmark}\]\nMarker (·) used to mark write operations with `debug=showwrgloss`.

\[\texttt{\hyperbf\{}\langle\text{location(s)}\rangle\}\]\nIf hyperlinks are supported this does `\textbf{\glshypernumber\{}\langle\text{location(s)}\rangle\}` otherwise it just does `\textbf{\langle\text{location}\rangle}`.

\[\texttt{\ifglossaryexists\{}\langle\text{glossary-type}\rangle\{\langle\text{true}\rangle\}{\langle\text{false}\rangle} \quad \text{modifier: *}\}\]\nIf the glossary given by `\langle\text{glossary-type}\rangle` exists, this does `\langle\text{true}\rangle`, otherwise it does `\langle\text{false}\rangle`. The unstarred form treats ignored glossaries as non-existent. The starred form (v4.46+) will do `\langle\text{true}\rangle` if `\langle\text{glossary-type}\rangle` matches an ignored glossary.
Command Summary

\ifglsentryexists{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}  
glossaries

Does ⟨true⟩ if the entry given by ⟨entry-label⟩ exists, otherwise does ⟨false⟩.

\ifglsfieldcseq{⟨entry-label⟩}{⟨field-label⟩}{⟨cs-name⟩}{⟨true⟩}{⟨false⟩}  
glossaries v4.16+

Tests if the value of the given field is equal to the replacement text of the command given by the control sequence name ⟨cs-name⟩ using etoolbox’s \ifcsstrequal. Triggers an error if the given field (identified by its internal field label) hasn’t been defined. Uses \glsdoifexists.

\ifglsfielddefeq{⟨entry-label⟩}{⟨field-label⟩}{⟨cs⟩}{⟨true⟩}{⟨false⟩}  
glossaries v4.16+

Tests if the value of the given field is equal to the replacement text of the given command ⟨cs⟩ using etoolbox’s \ifdefstrequal. Triggers an error if the given field (identified by its internal field label) hasn’t been defined. Uses \glsdoifexists.

\ifglsfieldeq{⟨entry-label⟩}{⟨field-label⟩}{⟨string⟩}{⟨true⟩}{⟨false⟩}  
glossaries v4.16+

Tests if the value of the given field is equal to the given string using etoolbox’s \ifcsstring. Triggers an error if the given field (identified by its internal field label) hasn’t been defined. Uses \glsdoifexists.

\ifglsfieldvoid{⟨field-label⟩}{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}  
glossaries v4.50+

An expandable test to determine if the entry is undefined or the field is undefined or empty. The ⟨field-label⟩ must be the field’s internal label.

\ifglshaschildren{⟨entry-label⟩}{⟨true⟩}{⟨false⟩}  
glossaries v3.02+
Command Summary

Does \langle true\rangle if the given entry has child entries otherwise does \langle false\rangle. Note that this has to iterate over the set of defined entries for the entry’s glossary to find one that has the entry identified in its parent field. A more efficient approach can be achieved with bib2gls and the save-child-count resource option.

\texttt{\textbackslash ifglshasdesc\{(entry-label)\}\{\{true\}\}\{\{false\}\}} \hfill \textit{glossaries v3.08a+}

Does \langle true\rangle if the entry’s description field is set otherwise does \langle false\rangle.

\texttt{\textbackslash ifglshasdescsuppressed\{(entry-label)\}\{\{true\}\}\{\{false\}\}} \hfill \textit{glossaries v3.08a+}

Does \langle true\rangle if the entry’s description field is just \texttt{\nopostdesc} otherwise does \langle false\rangle.

\texttt{\textbackslash ifglshasfield\{(field)\}\{\{entry-label\}\}\{\{true\}\}\{\{false\}\}} \hfill \textit{glossaries v4.03+}

If the field identified by either its key or its internal field label \langle field\rangle for the entry identified by \langle entry-label\rangle is set and non-empty, this sets \texttt{\glscurrentfieldvalue} to the field value and does \langle true\rangle otherwise it does \langle false\rangle.

\texttt{\textbackslash ifglshaslong\{entry-label\}\{\{true\}\}\{\{false\}\}} \hfill \textit{glossaries v3.11a+}

Does \langle true\rangle if the entry’s long field is set otherwise does \langle false\rangle.

\texttt{\textbackslash ifglshasparent\{entry-label\}\{\{true\}\}\{\{false\}\}} \hfill \textit{glossaries v3.02+}

Does \langle true\rangle if the entry’s parent field is set otherwise does \langle false\rangle.

\texttt{\textbackslash ifglshasshort\{entry-label\}\{\{true\}\}\{\{false\}\}} \hfill \textit{glossaries v3.11a+}

Does \langle true\rangle if the entry’s short field is set otherwise does \langle false\rangle.
Command Summary

$\texttt{\ifglshassymbol\{entry-label\}\{true\}\{false\}}$

Does $\langle true \rangle$ if the entry’s symbol field is set otherwise does $\langle false \rangle$.

$\texttt{\ifglsindexonlyfirst \{true\}\else \{false\}\fi}$

A conditional that corresponds to the indexonlyfirst option.

$\texttt{\ifGlsLongExtraUseTabular \{true\}\else \{false\}\fi}$

Conditional that determines whether or not to use tabular instead of longtable. If this conditional is changed, it must be changed before the style is set.

$\texttt{\ifglsnogroupskip \{true\}\else \{false\}\fi}$

Conditional set by the nogroupskip option.

$\texttt{\ifglsresetcurrcount \{true\}\else \{false\}\fi}$

Conditional that determines whether or not to reset the entry counter to 0 when the first use flag is reset.

$\texttt{\ifglsused\{entry-label\}\{true\}\{false\}}$

Does $\langle true \rangle$ if the entry has been marked as used, does $\langle false \rangle$ if the entry is marked as unused, and does neither if the entry hasn’t been defined (but will generate an error or warning according to undefaction).

$\texttt{\ifglsxtrinitwrglossbefore \{true\}\else \{false\}\fi}$

§5.8; 268

§8.7.2; 451

§6.1; 311

§5.1.2; 194
A conditional that indicates whether or not \texttt{wrgloss=before} is set.

\texttt{\ifglsxtrinsertinside \langle true \rangle \else \langle false \rangle \fi}
\texttt{ initial: \iffalse}

A conditional used by the predefined abbreviation styles to determine whether the \langle \textit{insert} \rangle part should go inside or outside of the style’s font formatting commands.

\texttt{\ifGlsXtrPrefixLabelFallbackLast \langle true \rangle \else \langle false \rangle \fi initial: \iftrue}

Conditional that determines whether or not to use the last label prefix as the default.

\texttt{\ifglsxtrprintglossflatten \langle true \rangle \else \langle false \rangle \fi initial: \iffalse}

Conditional set by the \texttt{flatten} option.

\texttt{\ifmglsused \langle multi-label \rangle \{ \langle true \rangle \} \{ \langle false \rangle \}}
\texttt{ initial: \iffalse}

Does \langle true \rangle if the given multi-entry has been marked as used, otherwise does \langle false \rangle.

\texttt{\ifmultiglossaryentryglobal \langle true \rangle \else \langle false \rangle \fi initial: \iffalse}

If true, subsequent multi-entry definitions will be global.

\texttt{\IfTeXParserLib \{ \langle \LaTeX parser lib code \rangle \} \{ \langle \LaTeX code \rangle \}}
\texttt{ initial: \iffalse}

Defined by glossaries-extra-bib2gls to \langle \LaTeX code \rangle but defined by bib2gls’s interpreter to expand to \langle \LaTeX parser lib code \rangle.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\IN</td>
<td>Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \IN.</td>
</tr>
<tr>
<td>\indexname</td>
<td>Expands to the index title.</td>
</tr>
<tr>
<td>\INTERPRET</td>
<td>Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \INTERPRET.</td>
</tr>
<tr>
<td>\Iota</td>
<td>Defined with \providecommand, this just does \text{I}.</td>
</tr>
<tr>
<td>\Kappa</td>
<td>Defined with \providecommand, this just does \text{K}.</td>
</tr>
<tr>
<td>\LABELIFY</td>
<td>Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \LABELIFY.</td>
</tr>
<tr>
<td>\LABELIFYLIST</td>
<td>Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \LABELIFYLIST.</td>
</tr>
</tbody>
</table>
Command Summary

\texttt{\LC}

Defined by \texttt{\GlsXtrResourceInitEscSequences} to expand to detokenized \texttt{\LC}.

\texttt{\LEN}

Defined by \texttt{\GlsXtrResourceInitEscSequences} to expand to detokenized \texttt{\LEN}.

\texttt{\letabbreviationstyle{\langle new style\rangle}{\langle existing style\rangle}}
glossaries-extra v1.04+

Defines a synonym for an existing abbreviation style.

\texttt{\loadglsentries[\langle type\rangle]{\langle filename\rangle}}
glossaries

Locally assigns \texttt{\glsdefaulttype} to \texttt{\langle type\rangle} and inputs \texttt{\langle filename\rangle}. If the optional argument is omitted, the default glossary is assumed. Note that if any entries with \texttt{\langle filename\rangle} have the \texttt{type} key set (including implicitly in commands like \texttt{\newabbreviation}), then this will override the type given in the optional argument.

\texttt{\longnewglossaryentry{\langle entry-label\rangle}{\langle key=value list\rangle}{\langle description\rangle}}
glossaries v3.11a

Defines a new glossary entry with the given label. The second argument is a comma-separated list of glossary entry keys. The third argument is the description, which may include paragraph breaks.

\texttt{\longnewglossaryentry*[\langle entry-label\rangle}{\langle key=value list\rangle}{\langle description\rangle}}
glossaries-extra v1.12+

Like the unstarrered \texttt{\longnewglossaryentry} but doesn’t add the \texttt{\glsxtrpostlongdescription} hook.
Command Summary

M

\texttt{\MakeAcronymsAbbreviations}

Counteracts \texttt{\RestoreAcronyms}. Not recommended.

\texttt{\makefirstuc\{\langle text\rangle\}} \hspace{1cm} \texttt{mfirstuc}

Robust command that converts the first character of \texttt{\langle text\rangle} to uppercase unless \texttt{\langle text\rangle} starts with a command, in which case it will attempt to apply the case change to the first character of the first argument following the command, if the command is followed by a group. As from mfirstuc v2.08, this command internally uses \texttt{\MFUsentencecase} to perform the actual case-change. See the mfirstuc documentation for further details, either:

\texttt{texdoc mfirstuc}

or visit ctan.org/pkg/mfirstuc.

\texttt{\makeglossaries\[\langle types\rangle\]} \hspace{1cm} \texttt{glossaries}

Opens the associated glossary files that need to be processed by \texttt{makeindex} or \texttt{xindy}. The optional argument is only available with glossaries-extra and is used for a hybrid approach. All glossaries (or each glossary identified in \texttt{\langle types\rangle}) should be displayed with \texttt{\printglossary}. If the optional argument is present, any glossaries not identified in \texttt{\langle types\rangle} should be displayed with \texttt{\printnoidxglossary}.

\texttt{\makenoidxglossaries} \hspace{1cm} \texttt{glossaries v4.04+}

Sets up all non-ignored glossaries so that they can be displayed with \texttt{\printnoidxglossary}.

\texttt{\mfirstucMakeUppercase\{\langle text\rangle\}} \hspace{1cm} \texttt{mfirstuc}

This command was used by \texttt{\makefirstuc} to convert its argument to all caps and was redefined by glossaries to use \texttt{\MakeTextUppercase}, but with mfirstuc v2.08+ and glossaries
v4.50+ this command is instead defined to use the \TeX all caps command, which is expandable. This command is no longer used by \texttt{\makefirstuc} (which instead uses \texttt{\MFUsentencase}) or by glossaries v4.50+ (which now uses \texttt{\glsuppercase} for all caps commands such as \texttt{\GLS}).

\begin{verbatim}
\MFUaddmap{⟨cs1⟩}{⟨cs2⟩}
\end{verbatim} \quad \texttt{mfirstuc v2.08+}

Identifies a mapping from the command ⟨cs1⟩ to command ⟨cs2⟩ for \texttt{\makefirstuc} and also identifies ⟨cs2⟩ as a blocker. Mappings and blockers aren’t supported by \texttt{\MFUsentencase}, so both ⟨cs1⟩ and ⟨cs2⟩ are identified as exclusions for \texttt{\MFUsentencase}.

\begin{verbatim}
\MFUblocker{⟨cs⟩}
\end{verbatim} \quad \texttt{mfirstuc v2.08+}

Locally identifies ⟨cs⟩ as a blocker command for \texttt{\makefirstuc} and an exclusion for \texttt{\MFUsentencase} (which doesn’t support blockers).

\begin{verbatim}
\MFUexcl{⟨cs⟩}
\end{verbatim} \quad \texttt{mfirstuc v2.08+}

Locally identifies ⟨cs⟩ as an exclusion command, which will be recognised by both \texttt{\makefirstuc} and \texttt{\MFUsentencase}.

\begin{verbatim}
\MFUsave
\end{verbatim} \quad \texttt{mfirstuc v2.08+}

Saves the list of exclusions, blockers and mappings to the aux file (if required by some external tool, such as \texttt{bib2gls}). This command sets itself to \texttt{\relax} so it doesn’t repeat the action if used multiple times, and counteracts any use of \texttt{\MFUsaveatend}.

\begin{verbatim}
\MFUsaveatend
\end{verbatim} \quad \texttt{mfirstuc v2.08+}

Saves the list of exclusions, blockers and mappings to the aux file (if required by some external tool, such as \texttt{bib2gls}) at the end of the document. This command sets itself to \texttt{\relax} so it doesn’t repeat the action if used multiple times, but it can be overridden by \texttt{\MFUsave}.

\begin{verbatim}
\MFUsentencase{⟨text⟩}
\end{verbatim} \quad \texttt{mfirstuc v2.08+}
Command Summary

Fully expands \textit{text} and converts the first letter to uppercase. Unlike \texttt{\makefirstuc}, this command is expandable, but only recognises commands identified as exclusions. See the \texttt{mfirstuc} documentation for further details. This command is provided by \texttt{glossaries-extra v1.49+} if an old version of \texttt{mfirstuc} is detected.

\texttt{\textbar{}MGP}\texttt{\textbar{}}§11.5.2; 561

Defined by \texttt{\GlsXtrResourceInitEscSequences} to expand to detokenized \texttt{\textbar{}MGP}. Note that this isn’t the same as \texttt{\glscapturedgroup}.

\texttt{\textbar{}MGLS\{\texttt{(options)}\}\{(multi-label)\}\{(insert)\} modifiers: * + \{alt-mod\}}\texttt{\textbar{}}\texttt{\textbar{}}§7.11.1; 367

As \texttt{\mgls} but uses \texttt{\GLS} for each element.

\texttt{\textbar{}MGlS\{\texttt{(options)}\}\{(multi-label)\}\{(insert)\} modifiers: * + \{alt-mod\}}\texttt{\textbar{}}\texttt{\textbar{}}§7.11.1; 366

As \texttt{\mgls} but uses \texttt{\Gls} for all elements.

\texttt{\textbar{}Mgls\{\texttt{(options)}\}\{(multi-label)\}\{(insert)\} modifiers: * + \{alt-mod\}}\texttt{\textbar{}}\texttt{\textbar{}}§7.11.1; 366

As \texttt{\mgls} but uses \texttt{\Gls} for the first element.

\texttt{\textbar{}mgls\{\texttt{(options)}\}\{(multi-label)\}\{(insert)\} modifiers: * + \{alt-mod\}}\texttt{\textbar{}}\texttt{\textbar{}}§7; 326

References a multi-entry identified by the given \texttt{(multi-label)}.

\texttt{\textbar{}mglsAddOptions\{\texttt{(multi-label)}\}\{(new-options)\}}\texttt{\textbar{}}\texttt{\textbar{}}§7; 328

Locally appends to the options associated with the given multi-entry.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mglscurrentcategory</td>
<td>Placeholder command for use in multi-entry hooks, this expands to the multi-entry category current in effect.</td>
</tr>
<tr>
<td>\mglscurrentlabel</td>
<td>Placeholder command for use in multi-entry hooks, this expands to the current element label.</td>
</tr>
<tr>
<td>\mglscurrentlist</td>
<td>Placeholder command for use in multi-entry hooks, this expands to the complete comma-separated list of elements.</td>
</tr>
<tr>
<td>\mglscurrentmainlabel</td>
<td>Placeholder command for use in multi-entry hooks, this expands to the label of the main element.</td>
</tr>
<tr>
<td>\mglscurrentmultilabel</td>
<td>Placeholder command for use in multi-entry hooks, this expands to the multi-entry label.</td>
</tr>
<tr>
<td>\mglscurrentoptions</td>
<td>Placeholder command for use in multi-entry hooks, this expands to the options used when the multi-entry was defined.</td>
</tr>
<tr>
<td>\mglscurrentprefix</td>
<td>Placeholder command for use in multi-entry hooks, this expands to the current prefix.</td>
</tr>
</tbody>
</table>
Command Summary

\texttt{\textbackslash mglscurrentsuffix} \texttt{\textbackslash glossaries\textbackslash extra v1.48+} \texttt{§7.5; 351}

Placeholder command for use in multi-entry hooks, this expands to the current suffix.

\texttt{\textbackslash mglscustompostlinkhook} \texttt{\textbackslash glossaries\textbackslash extra v1.48+} \texttt{§7.6; 352}

Hook used with the \texttt{mpostlinkelement=custom} option.

\texttt{\textbackslash mglselementindex} \texttt{\textbackslash glossaries\textbackslash extra v1.48+} \texttt{§7.5; 351}

A count register used in multi-entry hooks, this is set to the element index.

\texttt{\textbackslash mglselementposthook} \texttt{\textbackslash glossaries\textbackslash extra v1.48+} \texttt{§7.5; 350}

Hook performed after each (non-skipped) element in a multi-entry set.

\texttt{\textbackslash mglselementprehook} \texttt{\textbackslash glossaries\textbackslash extra v1.48+} \texttt{§7.5; 350}

Hook performed before each (non-skipped) element in a multi-entry set.

\texttt{\textbackslash mglselementreset\{entry-label\}} \texttt{\textbackslash glossaries\textbackslash extra v1.48+} \texttt{§7.10; 365}

945
Command Summary

Used by options such as `resetall` to reset an element’s first use flag (taking the `preset-local` option into account).

\[\texttt{\textbackslash mglselementunset\{\langle entry-label\rangle\}}\]  
\[\text{glossaries-extra v1.48+}\]  
\[\text{§7.10; 365}\]

Used by options such as `unsetall` to unset an element’s first use flag (taking the `preset-local` option into account).

\[\texttt{\textbackslash mglselementunset\{\langle entry-label\rangle\}}\]  
\[\text{glossaries-extra v1.48+}\]  
\[\text{§7.10; 365}\]

Expands to the internal field label required by `\mglsusefield`.

\[\texttt{\textbackslash mglsfield}\]  
\[\textit{initial: useri}\]  
\[\text{glossaries-extra v1.48+}\]  
\[\text{§7.11.3; 369}\]

Iterates over the list of element labels for the multi-entry identified by `<multi-label>`.

\[\texttt{\textbackslash mglsforeach\{\langle multi-label\rangle\}\{\langle cs\rangle\}\{\langle body\rangle\}}\]  
\[\text{glossaries-extra v1.48+}\]  
\[\text{§7.13; 373}\]

As `\mglsforeach` but skips the main entry label.

\[\texttt{\textbackslash mglsforetherelements\{\langle multi-label\rangle\}\{\langle cs\rangle\}\{\langle body\rangle\}}\]  
\[\text{glossaries-extra v1.48+}\]  
\[\text{§7.13; 374}\]

As `\mglsfull` but uses `\glsxtrfull` for any elements that have the `short` field set and `\glsfirst` otherwise.

\[\texttt{\textbackslash mglsfull\{\langle options\rangle\}\{\langle multi-label\rangle\}\{\langle insert\rangle\}}\]  
\[\text{modifiers: } * + \langle alt-mod\rangle\]  
\[\text{glossaries-extra v1.48+}\]  
\[\text{§7.11.2; 368}\]

As `\mglsfull` but sentence case.

\[\texttt{\textbackslash mglsfull\{\langle options\rangle\}\{\langle multi-label\rangle\}\{\langle insert\rangle\}}\]  
\[\text{modifiers: } * + \langle alt-mod\rangle\]  
\[\text{glossaries-extra v1.48+}\]  
\[\text{§7.11.2; 367}\]
Command Summary

\mglshascategoryprefix{\langle category-label\rangle}{\langle true\rangle}{\langle false\rangle} glossaries-extra v1.48+

§7.3; 345

Does \langle true\rangle if the given multi-entry category has a prefix set otherwise does \langle false\rangle.

\mglshascategoriesuffix{\langle category-label\rangle}{\langle true\rangle}{\langle false\rangle} glossaries-extra v1.48+

§7.3; 345

Does \langle true\rangle if the given multi-entry category has a suffix set otherwise does \langle false\rangle.

\mglsiflast{\langle true\rangle}{\langle false\rangle}

glossaries-extra v1.48+

§7.5; 352

For use in multi-entry hooks, this expands to \langle true\rangle if this is the last iteration otherwise expands to \langle false\rangle.

\mglsiflastelementcapscase{\langle no-change\rangle}{\langle firstuc\rangle}{\langle all caps\rangle}

§7.6.1; 354

glossaries-extra v1.48+

For use in multi-entry suffix and post-link hooks, this expands to the \langle no-change\rangle if the last element had no case-change applied, to \langle firstuc\rangle if the last element had sentence case applied or to \langle all caps\rangle if the last element had all caps applied.

\mglsiflastelementskipped{\langle true\rangle}{\langle false\rangle}

glossaries-extra v1.48+

§7.6.1; 353

For use in multi-entry suffix and post-link hooks, this expands to the \langle true\rangle if the last element was skipped, otherwise to \langle false\rangle.

\mglsiflastelementwastfirstuse{\langle true\rangle}{\langle false\rangle}

glossaries-extra v1.48+

§7.6.1; 353

For use in multi-entry suffix and post-link hooks, this expands to the \langle true\rangle if the last element was used for the first time, otherwise to \langle false\rangle.

\mglsiflastelementwaspplural{\langle true\rangle}{\langle false\rangle}

glossaries-extra v1.48+

§7.6.1; 354

947
Command Summary

For use in multi-entry suffix and post-link hooks, this expands to the \langle true\rangle if the last element had the plural form displayed, otherwise to \langle false\rangle.

\texttt{\textbackslash mglsiflastmaincapscase\{\langle no-change\rangle\}{\langle firstuc\rangle}{\langle all caps\rangle}}
glossaries-extra v1.48+

For use in multi-entry suffix and post-link hooks, this expands to the \langle no-change\rangle if the main element from the multi-entry just referenced had no case-change applied, to \langle firstuc\rangle if the last element had sentence case applied or to \langle all caps\rangle if the last element had all caps applied.

\texttt{\textbackslash mglsiflastmainskipped\{\langle true\rangle\}{\langle false\rangle}}
glossaries-extra v1.48+

For use in multi-entry suffix and post-link hooks, this expands to the \langle true\rangle if the main element from the multi-entry just referenced was skipped, otherwise to \langle false\rangle.

\texttt{\textbackslash mglsiflastmainwasfirstuse\{\langle true\rangle\}{\langle false\rangle}}
glossaries-extra v1.48+

For use in multi-entry suffix and post-link hooks, this expands to the \langle true\rangle if the main element from the multi-entry just referenced was used for the first time, otherwise to \langle false\rangle.

\texttt{\textbackslash mglsiflastmainwasplural\{\langle true\rangle\}{\langle false\rangle}}
glossaries-extra v1.48+

For use in multi-entry suffix and post-link hooks, this expands to the \langle true\rangle if the main element from the multi-entry just referenced had the plural form shown, otherwise to \langle false\rangle.

\texttt{\textbackslash mglsisfirstuse\{\langle true\rangle\}{\langle false\rangle}}
glossaries-extra v1.48+

For use in multi-entry hooks, this expands to \langle true\rangle if this is the first use otherwise expands to \langle false\rangle.

\texttt{\textbackslash mglslastcategory}
glossaries-extra v1.48+

§7.5; 351

§7.6; 353

§7.6.2; 354

§7.6.2; 355
For use in multi-entry suffix and post-link hooks, this expands to the multi-entry category or nothing, if no category assigned.

\texttt{\textbackslash mgslastelementlabel} \hspace{1cm} \texttt{\textbackslash glossaries-extra v1.48+} \hspace{1cm} \textsection{7.6.1; 353}

For use in multi-entry suffix and post-link hooks, this expands to the label of the last non-skipped element.

\texttt{\textbackslash mgslastelementpostlinkhook} \hspace{1cm} \texttt{\textbackslash glossaries-extra v1.48+} \hspace{1cm} \textsection{7.6; 352}

Hook used with the \texttt{mpostlinkelement=last} option.

\texttt{\textbackslash mgslastmainlabel} \hspace{1cm} \texttt{\textbackslash glossaries-extra v1.48+} \hspace{1cm} \textsection{7.6.2; 354}

For use in multi-entry suffix and post-link hooks, this expands to the label of the main element that was just referenced.

\texttt{\textbackslash mgslastmainpostlinkhook} \hspace{1cm} \texttt{\textbackslash glossaries-extra v1.48+} \hspace{1cm} \textsection{7.6; 352}

Hook used with the \texttt{mpostlinkelement=main} option.

\texttt{\textbackslash mgslastmultilabel} \hspace{1cm} \texttt{\textbackslash glossaries-extra v1.48+} \hspace{1cm} \textsection{7.6; 353}

For use in multi-entry suffix and post-link hooks, this expands to the multi-entry label.

\texttt{\textbackslash mglslocalreset\{\textsl{\textbackslash multi-label}\}} \hspace{1cm} \texttt{\textbackslash glossaries-extra v1.48+} \hspace{1cm} \textsection{7.7; 356}

Locally resets the first use flag for the given multi-entry.

\texttt{\textbackslash mglslocalunset\{\textsl{\textbackslash multi-label}\}} \hspace{1cm} \texttt{\textbackslash glossaries-extra v1.48+} \hspace{1cm} \textsection{7.7; 356}
Command Summary

Locally unsets the first use flag for the given multi-entry.

\texttt{\textbackslash mglslunsetothers\{\langle multi-label\rangle\}} \hspace{1cm} \textit{glossaries-extra v1.48+} \hspace{1cm} \textsuperscript{\S 7.10; 365}

Locally unsets the first use flag for the other (not main) elements of the given multi-entry.

\texttt{\textbackslash Mglslong\{\langle options\rangle\}\{\langle multi-label\rangle\}\{\langle insert\rangle\}} \hspace{1cm} \textit{modifiers: \texttt{* + \textbackslash alt-mod}} \hspace{1cm} \textsuperscript{\S 7.11.2; 368}

As \texttt{\textbackslash mglslong} but sentence case.

\texttt{\textbackslash mglslong\{\langle options\rangle\}\{\langle multi-label\rangle\}\{\langle insert\rangle\}} \hspace{1cm} \textit{modifiers: \texttt{* + \textbackslash alt-mod}} \hspace{1cm} \textsuperscript{\S 7.11.2; 367}

As \texttt{\textbackslash mglslong} but uses \texttt{\textbackslash glsxtrlong} for any elements that have the \texttt{long} field set and \texttt{\textbackslash glstext} otherwise.

\texttt{\textbackslash MGLSmainpl\{\langle options\rangle\}\{\langle multi-label\rangle\}\{\langle insert\rangle\}} \hspace{1cm} \textit{modifiers: \texttt{* + \textbackslash alt-mod}} \hspace{1cm} \textsuperscript{\S 7.11.1; 367}

As \texttt{\textbackslash mglsp} but uses \texttt{\textbackslash GLSp1} for the main element and \texttt{\textbackslash GLS} for the others.

\texttt{\textbackslash MGLSmainpl\{\langle options\rangle\}\{\langle multi-label\rangle\}\{\langle insert\rangle\}} \hspace{1cm} \textit{modifiers: \texttt{* + \textbackslash alt-mod}} \hspace{1cm} \textsuperscript{\S 7.11.1; 367}

As \texttt{\textbackslash mglsp} but uses \texttt{\textbackslash Glsp1} for the main entry and \texttt{\textbackslash Gl} for the others.

\texttt{\textbackslash Mglsmainpl\{\langle options\rangle\}\{\langle multi-label\rangle\}\{\langle insert\rangle\}} \hspace{1cm} \textit{modifiers: \texttt{* + \textbackslash alt-mod}} \hspace{1cm} \textsuperscript{\S 7.11.1; 366}

As \texttt{\textbackslash mglsp} uses sentence case for the first element and the plural form for the main element.
As `\mgls` but uses the plural form for the main element.

As `\mgs` but uses `\Glsname`.

As `\mgs` but uses `\Glsname` for the first entry and `\glsname` for the remaining entries.

As `\mgs` but uses `\glsname`.

As `\mglspl` but uses `\GLSpl` for each element.

As `\mglspl` but uses `\Glspl` for each element.
Command Summary

As `\mglsp` but uses `\Glspl` for the first element and `\glspl` for the remaining elements.

\[\mglsp\{⟨options⟩\}{⟨multi-label⟩}\{⟨insert⟩\}\text{ modifiers: }* + ⟨alt-mod⟩\]
§7.11.1; 366

glossaries-extra v1.48+

As `\mgs` but uses the plural form for each element.

\mglsprefix
§7.3; 343

glossaries-extra v1.48+

Code used to typeset the multi-entry prefix.

\mglssreset{⟨multi-label⟩}
§7.7; 355

glossaries-extra v1.48+

Globally resets the first use flag for the given multi-entry.

\mglssresetall
§7.7; 356

glossaries-extra v1.48+

Resets the first use flag for all multi-entries.

\mglssseefirstitem{⟨multi-label⟩}
§7.12; 372

glossaries-extra v1.48+

Formatting command used by cross-reference lists for the first item if the item is a multi-entry.

\mglssseeitem{⟨multi-label⟩}
§7.12; 372

glossaries-extra v1.48+

Formatting command used by cross-reference lists for subsequent items if the item is a multi-entry.

\mglssetmain{⟨multi-label⟩}{⟨new-main-label⟩}
§7.12; 372

glossaries-extra v1.48+
Command Summary

Locally changes the main element for the given multi-entry.

\[\text{\texttt{mglsSetOptions}}\{\langle \text{multi-label} \rangle}\{\langle \text{new-options} \rangle}\]  
\[\text{glossaries-extra v1.48+} \quad §7; 328\]

Locally sets the options associated with the given multi-entry.

\[\text{\texttt{Mglsshort}}\{\langle \text{options} \rangle\}\{\langle \text{multi-label} \rangle\}\{\langle \text{insert} \rangle\]  
\[\text{modifiers: } * + \langle \text{alt-mod} \rangle\]  
\[\text{glossaries-extra v1.48+} \quad §7.11.2; 368\]

As \texttt{mglsshort} but sentence case.

\[\text{\texttt{mglsshort}}\{\langle \text{options} \rangle\}\{\langle \text{multi-label} \rangle\}\{\langle \text{insert} \rangle\]  
\[\text{modifiers: } * + \langle \text{alt-mod} \rangle\]  
\[\text{glossaries-extra v1.48+} \quad §7.11.2; 367\]

As \texttt{mgls} but uses \texttt{glsxtrshort} for any elements that have the \texttt{short} field set and \texttt{gls-} 
\texttt{text} otherwise.

\[\text{\texttt{mglsuffix}}\]  
\[\text{glossaries-extra v1.48+} \quad §7.3; 344\]

Code used to typeset the multi-entry suffix.

\[\text{\texttt{MGlssymbol}}\{\langle \text{options} \rangle\}\{\langle \text{multi-label} \rangle\}\{\langle \text{insert} \rangle\]  
\[\text{modifiers: } * + \langle \text{alt-mod} \rangle\]  
\[\text{glossaries-extra v1.48+} \quad §7.11.3; 368\]

As \texttt{mgls} but uses \texttt{glssymbol} if the \texttt{symbol} field is set and \texttt{Gls} otherwise.

\[\text{\texttt{MGlssymbol}}\{\langle \text{options} \rangle\}\{\langle \text{multi-label} \rangle\}\{\langle \text{insert} \rangle\]  
\[\text{modifiers: } * + \langle \text{alt-mod} \rangle\]  
\[\text{glossaries-extra v1.48+} \quad §7.11.3; 369\]

As \texttt{mgls} but uses \texttt{glssymbol} if the \texttt{symbol} field is set, otherwise it uses \texttt{Gls} for the first 
\texttt{element} and \texttt{gls} for the remaining elements.

953
Command Summary

\mglssymbol\{\langle options\rangle\}\{\langle multi-label\rangle\}\{\langle insert\rangle\}  modifiers: * + \langle alt-mod\rangle

As \mglss but uses \glssymbol if the symbol field is set and \gls otherwise.

\mglsunset\{\langle multi-label\rangle\}  glossaries-extra v1.48+

Globally unsets the first use flag for the given multi-entry.

\mglsunsetall  glossaries-extra v1.48+

Unsets the first use flag for all multi-entries.

\mglsunsetothers\{\langle multi-label\rangle\}  glossaries-extra v1.48+

Globally unsets the first use flag for the other (not main) elements of the given multi-entry.

\mglssusecategoryprefix\{\langle category-label\rangle\}  glossaries-extra v1.48+

Expands to the prefix assigned to the given multi-entry category or does nothing if no prefix assigned.

\mglssusecategoriesuffix\{\langle category-label\rangle\}  glossaries-extra v1.48+

Expands to the suffix assigned to the given multi-entry category or does nothing if no suffix assigned.

\MGlsusefield\{\langle options\rangle\}\{\langle multi-label\rangle\}\{\langle insert\rangle\}  modifiers: * + \langle alt-mod\rangle

As \mglssusefield but sentence case for each element.
Command Summary

\texttt{\textbackslash Mglusefield}\{\langle\textbackslash options}\}\{\langle\textbackslash multi-label\rangle\}\{\langle\textbackslash insert\rangle\} \quad \text{modifiers: * + \langle\textbackslash alt-mod\rangle}

As \texttt{\textbackslash Mglusefield} but sentence case for the first element.

\texttt{\textbackslash Mglusefield} \{\langle\textbackslash options}\}\{\langle\textbackslash multi-label\rangle\}\{\langle\textbackslash insert\rangle\} \quad \text{modifiers: * + \langle\textbackslash alt-mod\rangle}

As \texttt{\textbackslash Mglusefield} but uses \texttt{\textbackslash glsdisp} if the field identified by \texttt{\textbackslash Mglfield} exists with the link text obtained from the field value.

\texttt{\textbackslash Mglswasfirstuse}\{\langle\textbackslash true\rangle\}\{\langle\textbackslash false\rangle\}

For use in multi-entry suffix and post-link hooks, this expands to the \{\langle\textbackslash true\rangle\} if this was the first use of the multi-entry, otherwise to \{\langle\textbackslash false\rangle\}.

\texttt{\textbackslash MPGLS}\{\langle\textbackslash options\rangle\}\{\langle\textbackslash multi-label\rangle\}\{\langle\textbackslash insert\rangle\} \quad \text{modifiers: * + \langle\textbackslash alt-mod\rangle}

As \texttt{\textbackslash MPGLS} but all caps for the all elements.

\texttt{\textbackslash MPGLs}\{\langle\textbackslash options\rangle\}\{\langle\textbackslash multi-label\rangle\}\{\langle\textbackslash insert\rangle\} \quad \text{modifiers: * + \langle\textbackslash alt-mod\rangle}

As \texttt{\textbackslash MPGLs} but sentence case for all elements.

\texttt{\textbackslash Mpgl}\{\langle\textbackslash options\rangle\}\{\langle\textbackslash multi-label\rangle\}\{\langle\textbackslash insert\rangle\} \quad \text{modifiers: * + \langle\textbackslash alt-mod\rangle}

As \texttt{\textbackslash Mpgl} but sentence case for the first element.
As \texttt{mgl}s but uses \texttt{pgl}s for the first element.

\begin{verbatim}
\texttt{MPGLSmainpl}[^{\langle\text{options}\rangle}\{\langle\text{multi-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle}
\end{verbatim}

\begin{verbatim}
glossaries-extra v1.48+
\end{verbatim}

§7.11.4; 372

As \texttt{mpgl}s\texttt{mainpl} but all caps for the all elements.

\begin{verbatim}
\texttt{MPGlsmainpl}[^{\langle\text{options}\rangle}\{\langle\text{multi-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle}
\end{verbatim}

\begin{verbatim}
glossaries-extra v1.48+
\end{verbatim}

§7.11.4; 371

As \texttt{mpgl}s\texttt{mainpl} but sentence case for all elements.

\begin{verbatim}
\texttt{Mpglsmainpl}[^{\langle\text{options}\rangle}\{\langle\text{multi-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle}
\end{verbatim}

\begin{verbatim}
glossaries-extra v1.48+
\end{verbatim}

§7.11.4; 371

As \texttt{mpgl}s\texttt{pl} but sentence case for the first element.

\begin{verbatim}
\texttt{mpglsmainpl}[^{\langle\text{options}\rangle}\{\langle\text{multi-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle}
\end{verbatim}

\begin{verbatim}
glossaries-extra v1.48+
\end{verbatim}

§7.11.4; 371

As \texttt{mgls} but uses \texttt{pgl} for the first element if its the main element otherwise \texttt{gls} and, for the remaining elements, uses \texttt{glspl} if the element is the main entry or \texttt{gls} otherwise.

\begin{verbatim}
\texttt{MPGLSpl}[^{\langle\text{options}\rangle}\{\langle\text{multi-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle}
\end{verbatim}

\begin{verbatim}
glossaries-extra v1.48+
\end{verbatim}

§7.11.4; 372

As \texttt{mpgl}pl but all caps for the all elements.

\begin{verbatim}
\texttt{MPGlspl}[^{\langle\text{options}\rangle}\{\langle\text{multi-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle}
\end{verbatim}

\begin{verbatim}
glossaries-extra v1.48+
\end{verbatim}

§7.11.4; 371

As \texttt{mpgl}pl but sentence case for all elements.
Command Summary

\texttt{\textbackslash mpglspl[@\langle options\rangle\{\langle multi-label\rangle\}\{\langle insert\rangle\}]\text{ modifiers: } * + \langle alt-mod\rangle\}\\glossaries-extra v1.48+}

As \texttt{\textbackslash mgs} but uses \texttt{\textbackslash pglspl} for the first element and \texttt{\textbackslash glsp} for the remaining elements.

\texttt{\textbackslash mpglsWarning}\glossaries-extra v1.48+

Issues a warning with \texttt{\GlossariesExtraWarning} indicating that glossaries-prefix is required for \texttt{\textbackslash mpgls} family of commands.

\texttt{\textbackslash Mu}\glossaries-extra-bib2gls v1.27+

Defined with \texttt{\providecommand}, this just does \texttt{\mathrm{M}}.

\texttt{\textbackslash multiglossaryentry[@\langle options\rangle\{\langle multi-label\rangle\}\{\langle main-label\rangle\}\{\langle entry-label-list\rangle\}]}\glossaries-extra v1.48+

Defines a multi-entry set with the label \langle multi-label\rangle, consisting of the entries whose labels are listed in \langle entry-label-list\rangle, where the main entry (which must be present in \langle entry-label-list\rangle) is identified by \langle main-label\rangle (or the final element in \langle entry-label-list\rangle, if \langle main-label\rangle is omitted).

\texttt{\textbackslash multiglossaryentryglobalfalse}\glossaries-extra v1.48+

Sets \texttt{\ifmultiglossaryentryglobal} to false.

\texttt{\textbackslash multiglossaryentryglobaltrue}\glossaries-extra v1.48+

Sets \texttt{\ifmultiglossaryentryglobal} to true.

\texttt{\textbackslash multiglossaryentrysetup\{\langle options\rangle\} }\glossaries-extra v1.48+

Specifies a general set of options to apply to all multi-entries.
Command Summary

N

\newabbr[⟨options⟩]{⟨entry-label⟩}{⟨short⟩}{⟨long⟩}

§4.3.2; Table 4.1

A synonym for \newabbreviation defined by the shortcuts=abbreviations or shortcuts =ac package option.

\newabbreviation[⟨options⟩]{⟨entry-label⟩}{⟨short⟩}{⟨long⟩}

§4.1; 41

Defines a new entry that represents an abbreviation. This internally uses \newglossary-entry and any provided ⟨options⟩ (glossary entry keys) will be appended. The category is set to abbreviation by default, but may be overridden in ⟨options⟩. The appropriate style should be set before the abbreviation is defined with \setabbreviationstyle.

\newabbreviationhook

§4.1.5; 45

Hook provided within \newabbreviation just before the entry is defined.

\newabbreviationstyle{⟨style-name⟩}{⟨setup⟩}{⟨display definitions⟩}

§4.5.3; 161

Defines an abbreviation style, which can be set with \setabbreviationstyle.

\newacronym[⟨options⟩]{⟨entry-label⟩}{⟨short⟩}{⟨long⟩}

This command is provided by the base glossaries package but is redefined by glossaries-extra to use \newabbreviation with the category key set to acronym. The appropriate style should be set before the abbreviation is defined with \setabbreviationstyle[acronym] {⟨style⟩}. You can override the category in ⟨options⟩ but remember to change the optional argument of \setabbreviationstyle to match.

\newacronymstyle{⟨name⟩}{⟨format def⟩}{⟨display defs⟩}

glossaries v4.02+

Defines an acronym style for use with the base glossaries package’s acronym mechanism. These styles are not compatible with glossaries-extra. Use \newabbreviationstyle instead.
Command Summary

\texttt{\newdglsfield[⟨default-options⟩]{⟨field⟩}{⟨cs⟩}} \hfill glossaries-extra-bib2gls v1.49+

§11.5.7; 592

Defines the command ⟨cs⟩[⟨options⟩]{⟨entry-label⟩} to behave like \texttt{\dglslfield[⟨default-options⟩,⟨options⟩]}{⟨entry-label⟩}{⟨field⟩}.

\texttt{\newdglsfieldlike[⟨default-options⟩]{⟨field⟩}{⟨cs⟩}{⟨Cs⟩}{⟨CS⟩}} \hfill glossaries-extra-bib2gls v1.49+

§11.5.7; 592

Similar to \texttt{\newdglsfield} but also defines sentence case (⟨Cs⟩) and all caps (⟨CS⟩) commands with mappings.

\texttt{\newentry{⟨entry-label⟩}{⟨options⟩}} \hfill §2.4; 18

A synonym for \texttt{\newglossaryentry} defined by the \texttt{shortcuts=other} package option.

\texttt{\newglossary[⟨log-ext⟩]{⟨glossary-label⟩}{⟨in-ext⟩}{⟨out-ext⟩}{⟨title⟩}} \hfill glossaries

[⟨counter⟩]

§11.5.7; 592

Defines a glossary identified by ⟨glossary-label⟩ (which can be referenced by the \texttt{type} key when defining an entry). The ⟨title⟩ will be used when displaying the glossary (using commands like \texttt{\printglossary}), but this title can be overridden by the \texttt{title} option. The optional ⟨counter⟩ indicates which counter should be used by default for the location when indexing any entries that have been assigned to this glossary. (This can be overridden by the \texttt{counter} option.) The other arguments are file extensions for use with \texttt{makeindex} or \texttt{xindy}. These arguments aren’t relevant for other indexing options (in which case, you may prefer to use \texttt{\newglossary*}).

\texttt{\newglossary*{⟨glossary-label⟩}{⟨title⟩}[⟨counter⟩]} \hfill glossaries v4.08+

A shortcut that supplies file extensions based on the glossary label:

\texttt{\newglossary[⟨glossary-label⟩-glg]{⟨glossary-label⟩}{⟨glossary-label⟩-gls}} {⟨⟨glossary-label⟩-glo⟩}{⟨title⟩}[⟨counter⟩]


\newglossaryentry{⟨entry-label⟩}{⟨key=value list⟩}

Defines a new glossary entry with the given label. The second argument is a comma-separated list of glossary entry keys.

\newignoredglossary{⟨glossary-label⟩}

Defines a glossary that should be ignored by iterative commands, such as \printglossaries. This glossary has no associated indexing files and has hyperlinks disabled. You can use an ignored glossary for common terms or abbreviations that don’t need to be included in any listing (but you may want these terms defined as entries to allow automated formatting with the \gls-like commands). An ignored glossary can’t be displayed with \printglossary but may be displayed with the “unsrt” family of commands, such as \printunsrtglossary.

\newignoredglossary*{⟨glossary-label⟩}

This is like the unstarred \newignoredglossary but doesn’t disable hyperlinks. You will need to ensure that the hypertargets are defined. For example, with \printunsrtglossary or through standalone entrys.

\newnum{⟨key=value list⟩}{⟨entry-label⟩}{⟨num⟩}

A synonym for \glsxtrnewnumber defined by the shortcuts=other package option (provided the numbers option is also used).

\newsym{⟨key=value list⟩}{⟨entry-label⟩}{⟨sym⟩}

A synonym for \glsxtrnewsymbol defined by the shortcuts=other package option (provided the symbols option is also used).

\newterm{⟨key=value list⟩}{⟨entry-label⟩}

(requires index package option)

(glossaries v4.02+)

§2.1; 12

§2.4; 18

§2.4; 19

§2.4; 19

§8; 378
Command Summary

Defines a new glossary entry with the given label, type set to index, the name set to \entrylabel{} and the description set to \nopostdesc. The optional argument is a comma-separated list of glossary entry keys, which can be used to override the defaults.

\NIN

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \NIN.

\nopostdesc glossaries v1.17+

When placed at the end of the description, this switches off the post-description hook (including the post-description punctuation). Does nothing outside of the glossary.

\NOTPREFIXOF

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \NOTPREFIXOF.

\NOTSUFFIXOF

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \NOTSUFFIXOF.

\Nu glossaries-extra-bib2gls v1.27+

Defined with \providecommand, this just does \mathrm{N}.

\NULL

Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \NULL.

O
Defined with `\providecommand`, this just does `\mathrm{O}`.

\begin{itemize}
  \item Defined with \providecommand, this just does \textit{mathrm{O}}.
\end{itemize}

\textbf{\texttt{P}}

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\pagelistname</td>
<td>initial: Page List glossaries (language-sensitive)</td>
</tr>
<tr>
<td>\PGLS{(options)}{⟨entry-label⟩}{⟨insert⟩}</td>
<td>modifiers: * + ⟨alt-mod⟩ glossaries-prefix</td>
</tr>
<tr>
<td>\Pgls{(options)}{⟨entry-label⟩}{⟨insert⟩}</td>
<td>modifiers: * + ⟨alt-mod⟩ glossaries-prefix</td>
</tr>
<tr>
<td>\pgls{(options)}{⟨entry-label⟩}{⟨insert⟩}</td>
<td>modifiers: * + ⟨alt-mod⟩ glossaries-prefix</td>
</tr>
<tr>
<td>\PGLSfmtlong{(options)}{⟨entry-label⟩}{⟨insert⟩}</td>
<td>modifiers: * + ⟨alt-mod⟩ glossaries-prefix</td>
</tr>
</tbody>
</table>

As \texttt{pgls} but all caps.

As \texttt{pgls} but sentence case.

Similar to \texttt{gls} but inserts the appropriate prefix, if provided.

As \texttt{pglsfmtlong} but all caps.
Command Summary

\Pglsmfitlong\[⟨options⟩\]{⟨entry-label⟩}{⟨insert⟩}

glossaries-extra v1.49+
(requires glossaries-prefix)

Modifiers: * + ⟨alt-mod⟩

As \pglsfmtlong but sentence case.

\pglsfmtlong\[⟨options⟩\]{⟨entry-label⟩}{⟨insert⟩}

glossaries-extra v1.49+

Modifiers: * + ⟨alt-mod⟩

As \glsfmtlong but inserts the prefixfield and separator in front if set.

\PGLSfmtlongpl\[⟨options⟩\]{⟨entry-label⟩}{⟨insert⟩}

glossaries-extra v1.49+
(requires glossaries-prefix)

Modifiers: * + ⟨alt-mod⟩

As \pglsfmtlongpl but all caps.

\PGLSfmtlongpl\[⟨options⟩\]{⟨entry-label⟩}{⟨insert⟩}

glossaries-extra v1.49+
(requires glossaries-prefix)

Modifiers: * + ⟨alt-mod⟩

As \pglsfmtlongpl but sentence case.

\pglsfmtlongpl\[⟨options⟩\]{⟨entry-label⟩}{⟨insert⟩}

glossaries-extra v1.49+
(requires glossaries-prefix)

Modifiers: * + ⟨alt-mod⟩

As \glsfmtlongpl but inserts the prefixfield and separator in front if set.

\PGLSfmtshort\[⟨options⟩\]{⟨entry-label⟩}{⟨insert⟩}

glossaries-extra v1.49+
(requires glossaries-prefix)

Modifiers: * + ⟨alt-mod⟩

As \pglsfmtshort but all caps.

As \pglsfmtshort but all caps.
**Command Summary**

\texttt{\Pglst}{\langle\text{options}\rangle}\{\langle\text{entry-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle

\text{glossaries-extra v1.49+}
\text{(requires glossaries-prefix)}

As \texttt{\pgls} but sentence case.

\texttt{\Pglst}{\langle\text{options}\rangle}\{\langle\text{entry-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle

\text{glossaries-extra v1.49+}
\text{(requires glossaries-prefix)}

As \texttt{\glst} but inserts the \texttt{prefix} field and separator in front if set.

\texttt{\PGLS}{\langle\text{options}\rangle}\{\langle\text{entry-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle

\text{glossaries-extra v1.49+}
\text{(requires glossaries-prefix)}

As \texttt{\glstpl} but all caps.

\texttt{\PGLS}{\langle\text{options}\rangle}\{\langle\text{entry-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle

\text{glossaries-extra v1.49+}
\text{(requires glossaries-prefix)}

As \texttt{\glstpl} but sentence case.

\texttt{\PGLS}{\langle\text{options}\rangle}\{\langle\text{entry-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle

\text{glossaries-extra v1.49+}
\text{(requires glossaries-prefix)}

As \texttt{\glstpl} but inserts the \texttt{prefixplural} field and separator in front if set.

\texttt{\PGLS}\{\langle\text{options}\rangle}\{\langle\text{entry-label}\rangle}\{\langle\text{insert}\rangle\} \quad \text{modifiers: } * + \langle\text{alt-mod}\rangle

\text{glossaries-prefix}

As \texttt{\pgls} but all caps.
### Command Summary

\`\texttt{\textbackslash Pglspl}[\langle options\rangle]\{\langle entry-label\rangle}\}[[\langle insert\rangle]]\]

**Modifiers:** * + \langle alt-mod\rangle

As \texttt{\textbackslash pgls} but sentence case.

\`\texttt{\textbackslash pglspl}[\langle options\rangle]\{\langle entry-label\rangle}\}[[\langle insert\rangle]]\]

**Modifiers:** * + \langle alt-mod\rangle

Similar to \texttt{\textbackslash glspl} but inserts the appropriate prefix, if provided.

\`\texttt{\textbackslash PGLSxtrlong}[\langle options\rangle]\{\langle entry-label\rangle}\}[[\langle insert\rangle]]\]

**Modifiers:** * + \langle alt-mod\rangle

\texttt{\textbackslash PGLSxtrlong} requires \texttt{glossaries-prefix}.

As \texttt{\textbackslash pglsxtrlong} but all caps.

\`\texttt{\textbackslash PGLSxtrlongpl}[\langle options\rangle]\{\langle entry-label\rangle}\}[[\langle insert\rangle]]\]

**Modifiers:** * + \langle alt-mod\rangle

\texttt{\textbackslash PGLSxtrlongpl} requires \texttt{glossaries-prefix}.

As \texttt{\textbackslash pglsxtrlongpl} but all caps.

§4.3.1; 56

As \texttt{\textbackslash pglsxtrlong} but inserts the \texttt{prefix} first field and separator in front if set.


\textbf{Command Summary}

\begin{itemize}
  \item \texttt{\textbackslash pglsxtrlongpl[\langle options\rangle][\langle entry-label\rangle][\langle insert\rangle]} \quad \text{modifiers:} \quad * + \langle alt-mod \rangle
  \text{glossaries-extra v1.49+}
  \text{(requires glossaries-prefix)}

  \text{As} \texttt{\textbackslash pglsxtrlongpl} \text{but sentence case.}

  \begin{itemize}
    \item \texttt{\textbackslash pglsxtrshort[\langle options\rangle][\langle entry-label\rangle][\langle insert\rangle]} \quad \text{modifiers:} \quad * + \langle alt-mod \rangle
      \text{glossaries-extra v1.49+}
      \text{(requires glossaries-prefix)}
    \end{itemize}

  \text{As} \texttt{\textbackslash glsxltrshortpl} \text{but inserts the prefixfirstplural field and separator in front if set.}

  \begin{itemize}
    \item \texttt{\textbackslash glsxltrshort[\langle options\rangle][\langle entry-label\rangle][\langle insert\rangle]} \quad \text{modifiers:} \quad * + \langle alt-mod \rangle
      \text{glossaries-extra v1.49+}
      \text{(requires glossaries-prefix)}
    \end{itemize}

  \text{As} \texttt{\textbackslash glsxltrshort} \text{but all caps.}

  \begin{itemize}
    \item \texttt{\textbackslash glsxltrshort[\langle options\rangle][\langle entry-label\rangle][\langle insert\rangle]} \quad \text{modifiers:} \quad * + \langle alt-mod \rangle
      \text{glossaries-extra v1.49+}
      \text{(requires glossaries-prefix)}
    \end{itemize}

  \text{As} \texttt{\textbackslash glsxltrshort} \text{but sentence case.}

  \begin{itemize}
    \item \texttt{\textbackslash glsxltrshort[\langle options\rangle][\langle entry-label\rangle][\langle insert\rangle]} \quad \text{modifiers:} \quad * + \langle alt-mod \rangle
      \text{glossaries-extra v1.49+}
      \text{(requires glossaries-prefix)}
    \end{itemize}

  \text{As} \texttt{\textbackslash glsxltrshort} \text{but inserts the prefix field and separator in front if set.}

  \begin{itemize}
    \item \texttt{\textbackslash glsxltrshortpl[\langle options\rangle][\langle entry-label\rangle][\langle insert\rangle]} \quad \text{modifiers:} \quad * + \langle alt-mod \rangle
      \text{glossaries-extra v1.49+}
      \text{(requires glossaries-prefix)}
    \end{itemize}
\end{itemize}
Command Summary

As \pglsxtrshortpl but all caps.

\texttt{\pglsxtrshortpl} [⟨options⟩] {⟨entry-label⟩} [⟨insert⟩]  
\texttt{modifers: * + (alt-mod)}  
glossaries-extra v1.49+  
(requires glossaries-prefix)  
§4.3.1; 55

As \pglsxtrshortpl but sentence case.

\texttt{\pglsxtrshortpl} [⟨options⟩] {⟨entry-label⟩} [⟨insert⟩]  
\texttt{modifers: * + (alt-mod)}  
glossaries-extra v1.49+  
(requires glossaries-prefix)  
§4.3.1; 55

As \glsxtrshortpl but inserts the prefixplural field and separator in front if set.

\texttt{\glsxtrtitlelong} {⟨entry-label⟩}  
glossaries-extra v1.49+  
(requires glossaries-prefix)  
§5.3.3; 219

The normal behaviour of \Pglsfmtlong.

\texttt{\glsxtrtitlelongpl} {⟨entry-label⟩}  
glossaries-extra v1.49+  
(requires glossaries-prefix)  
§5.3.3; 219

The normal behaviour of \Pglsfmtlongpl.

\texttt{\glsxtrtitleshort} {⟨entry-label⟩}  
glossaries-extra v1.49+  
(requires glossaries-prefix)  
§5.3.3; 219

The normal behaviour of \Pglsfmtshort.

\texttt{\glsxtrtitleshortpl} {⟨entry-label⟩}  
glossaries-extra v1.49+  
(requires glossaries-prefix)  
§5.3.3; 219
The normal behaviour of \Pglstshortpl.

\PREFIXOF

Defined by \GlstrInitEscSequences to expand to detokenized \PREFIXOF.

\pretoglossarypreamble\langle type\rangle\langle text\rangle
glossaries-extra v1.12+

Prepends (locally) \langle text\rangle to the preamble for the glossary identified by \langle type\rangle. If \langle type\rangle is omitted, \glsdefaulttype is assumed.

\printabbreviations\langle options\rangle
\usepackage[abbreviations]{glossaries-extra}

Shortcut for \printglossary[type=\glsxtrabbrvtype].

\printacronyms\langle options\rangle
glossaries

(requires the acronyms package option)

Shortcut for \printglossary[type=\acronymtype].

\printglossaries
glossaries

Iterates over all non-ignored glossaries and does \printglossary[type=\langle type\rangle] for each glossary.

\printglossary\langle options\rangle
glossaries

Displays the glossary by inputting a file created by makeindex or xindy. Must be used with \makeglossaries and either makeindex or xindy.
Command Summary

\printindex\[\langle\text{options}\rangle\] v4.02+  
(requires the \textit{index} package option)

Shortcut provided by the \textit{index} package option that simply does \texttt{\textbackslash printglossary[type=index]}.

\printnoidxglossaries  
glossaries v4.04+

Iterates over all non-ignored glossaries and does \texttt{\textbackslash printnoidxglossary[type=\langle\text{type}\rangle]} for each glossary.

\printnoidxglossary\[\langle\text{options}\rangle\]  
glossaries v4.04+

Displays the glossary by obtaining the indexing information from the aux file and using \TeX to sort and collate. Must be used with \texttt{\textbackslash makenoidxglossaries} or with the glossaries not identified in the optional argument of \texttt{\textbackslash makeglossaries} when using the hybrid method. This method can be very slow and has limitations.

\printnumbers\[\langle\text{options}\rangle\]  
glossaries v4.02+  
(requires the \textit{numbers} package option)

Shortcut for \texttt{\textbackslash printglossary[type=numbers]}.

\printsymbols\[\langle\text{options}\rangle\]  
glossaries v4.02+  
(requires the \textit{symbols} package option)

Shortcut for \texttt{\textbackslash printglossary[type=symbols]}.

\printunsrtabbreviations\[\langle\text{options}\rangle\]  
glossaries-extra-bib2gls v1.40+  
(requires \texttt{\usepackage[abbreviations,record]{glossaries-extra}})

Shortcut for \texttt{\textbackslash printunsrtglossary[type=\texttt{glsxtrabbvrtype}]}.

§11.5.1; 559

969
Command Summary

\printunsrtacronyms[(options)] \hspace{1cm} \text{glossaries-extra-bib2gls v1.40+}
\text{(requires \usepackage[acronyms,record]{glossaries-extra})}

§11.5.1; 559

Shortcut for \printunsrtglossary[type=\acronymtype].

\printunsrtglossaries \hspace{1cm} \text{glossaries-extra v1.08+}

§8.4; 387

Iterates over all non-ignored glossaries and does \printunsrtglossary[type=(type)] for each glossary.

\printunsrtglossary[(options)] \hspace{1cm} \text{glossaries-extra v1.08+}

§8.4; 386

Displays the glossary by iterating over all entries associated with the given glossary (in the order in which they were added to the glossary). Group headers will only be inserted if the group key has been defined and has been set (typically with the record option and bib2gls). Location lists will only be shown if the location or loclist fields have been set (typically by bib2gls).

\printunsrtglossary*[⟨options⟩]{⟨init-code⟩} \hspace{1cm} \text{glossaries-extra v1.12+}

§8.4; 386

Does ⟨init-code⟩\printunsrtglossary[(options)] which localises ⟨init-code⟩.

\printunsrtglossaryentryprocesshook{⟨entry-label⟩} \hspace{1cm} \text{glossaries-extra v1.21+}

§8.4.3; 406

Hook used within the “unsrt” family of commands while the glossary is being constructed.

\printunsrtglossarygrouphook{(internal cs)} \hspace{1cm} \text{glossaries-extra v1.50+}

§8.4.1; 396

Hook used within the “unsrt” family of commands while the group header is being constructed.

\printunsrtglossaryhandler{⟨entry-label⟩} \hspace{1cm} \text{glossaries-extra v1.16+}

§8.4.3; 407
Command Summary

Used within the “unsrt” family of commands to process the current entry.

\printunsrtglossarypostbegin\{⟨internal cs⟩\} glossaries-extra v1.50+ §8.4.3; 404

Hook used within the “unsrt” family of commands while the glossary is being constructed just after \begin{theglossary} is added.

\printunsrtglossarypostentryprocesshook\{⟨internal cs⟩\} glossaries-extra v1.50+ §8.4.3; 406

Hook used within the “unsrt” family of commands while the glossary is being constructed after the entry line has been added.

\printunsrtglossarypredoglossary glossaries-extra v1.21+ §8.4.3; 407

Hook performed by the “unsrt” family of commands just before the glossary body is displayed.

\printunsrtglossarypreend\{⟨internal cs⟩\} glossaries-extra v1.50+ §8.4.3; 404

Hook used within the “unsrt” family of commands while the glossary is being constructed just before \end{theglossary} is added.

\printunsrtglossarypreentryprocesshook\{⟨internal cs⟩\} glossaries-extra v1.50+ §8.4.3; 406

Hook used within the “unsrt” family of commands while the glossary is being constructed before the entry line has been added.

\printunsrtglossaryskipentry glossaries-extra v1.21+ §8.4.3; 406

May be used within \printunsrtglossaryentryprocesshook to skip the current entry.
Command Summary

\printunsrtglossaryunit[⟨options⟩]{⟨counter-name⟩}  
glossaries-extra v1.12+  
§8.4.3.2; 414

Provided for use with \GlsXtrRecordCounter to display a glossary with \printunsrtglossary* that filters entries that don’t have a match for the current ⟨counter-name⟩ value.

\printunsrtglossaryunitpostskip  
glossaries-extra v1.49+  
§8.4.3.2; 415

The vertical space at the end of the glossary appended by \printunsrtglossaryunit.

\printunsrtglossaryunitsetup{⟨counter-name⟩}  
glossaries-extra v1.12+  
§8.4.3.2; 414

Sets up the filtering used by \printunsrtglossaryunit.

\printunsrtindex[⟨options⟩]  
glossaries-extra-bib2gls v1.40+  
(requires \usepackage[index,record]{glossaries-extra})  
§11.5.1; 560

Shortcut provided by the index package option combined with glossaries-extra-bib2gls that simply does \printunsrtglossary[type=index].

\printunsrtinnerglossary[⟨options⟩]{⟨pre-code⟩}{⟨post-code⟩}  
glossaries-extra v1.44+  
§8.4.3.1; 409

Similar to \printunsrtglossary but doesn’t contain the code that starts and ends the glossary (such as beginning and ending the theglossary environment), so this command needs to be either placed inside printunsrtglossarywrap or in the \printunsrtglossary entry handler \printunsrtglossaryhandler.

\printunsrtnumbers[⟨options⟩]  
glossaries-extra-bib2gls v1.40+  
(requires \usepackage[numbers,record]{glossaries-extra})  
§11.5.1; 559

Shortcut provided by the numbers package option combined with glossaries-extra-bib2gls that simply does \printunsrtglossary[type=numbers].

972
Command Summary

\printunsrtsymbols[⟨options⟩]
  glossaries-extra-bib2gls v1.40+
  (requires \usepackage[symbols,record]{glossaries-extra})

Shortcut for \printunsrtglossary[type=symbols].

\printunsrttable[⟨options⟩]
  glossary-table v1.49+

Internally uses \printunsrtglossary with the table style.

\provideignoredglossary{⟨glossary-label⟩}
  modifier: *
  glossaries-extra v1.12+

As \newignoredglossary but does nothing if the glossary has already been defined.

\providemultiglossaryentry[⟨options⟩]{⟨multi-label⟩}{⟨main-label⟩}{⟨entry-label-list⟩}
  glossaries-extra v1.48+

As \multiglossaryentry but does nothing if a multi-entry set has already been defined with the given label.

\ProvidesGlossariesExtraLang{⟨tag⟩}

Should be placed at the start of a glossaries-extra .lfd file.

R

\renewabbreviationstyle{⟨style-name⟩}{⟨setup⟩}{⟨display definitions⟩}
  glossaries-extra v1.04+

Redefines an abbreviation style.

\RequireGlossariesExtraLang{⟨tag⟩}
Command Summary

Indicates that a glossaries-extra 1df file should be input, if it hasn’t already been input.

\RestoreAcronyms

$\S 4.6$; 181

Restores \texttt{\newacronym} to the base glossaries mechanism. Not recommended.

\rGLS\[\langle options\rangle\]\{\langle entry\label\rangle\}\[\langle insert\rangle\]

\textit{glossaries-extra v1.21+}

\textit{modifiers: * + \langle alt-mod\rangle}

$\S 11.4$; 551

Like \texttt{\GLS} but hooks into the entry’s record count.

\rGls\[\langle options\rangle\]\{\langle entry\label\rangle\}\[\langle insert\rangle\]

\textit{glossaries-extra v1.21+}

\textit{modifiers: * + \langle alt-mod\rangle}

$\S 11.4$; 551

Like \texttt{\Gls} but hooks into the entry’s record count.

\rgls\[\langle options\rangle\]\{\langle entry\label\rangle\}\[\langle insert\rangle\]

\textit{glossaries-extra v1.21+}

\textit{modifiers: * + \langle alt-mod\rangle}

$\S 11.4$; 550

Like \texttt{\gls} but hooks into the entry’s record count.

\rGLSformat\{\langle entry\label\rangle\}\{\langle insert\rangle\}

\textit{glossaries-extra v1.21+}

Format used by \texttt{\rGLS} if the entry’s record count is more than the given trigger value.

\rGlsformat\{\langle entry\label\rangle\}\{\langle insert\rangle\}

\textit{glossaries-extra v1.21+}

Format used by \texttt{\rGls} if the entry’s record count is more than the given trigger value.

\rglsformat\{\langle entry\label\rangle\}\{\langle insert\rangle\}

\textit{glossaries-extra v1.21+}

$\S 11.4$; 550

974
Format used by \texttt{\textbackslash rlgs} if the entry’s record count is more than the given trigger value.

\begin{verbatim}
\texttt{\textbackslash GLSpl}[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩}
\end{verbatim}

Like \texttt{\textbackslash GLSp1} but hooks into the entry’s record count.

\begin{verbatim}
\texttt{\textbackslash Glspl}[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩}
\end{verbatim}

Like \texttt{\textbackslash glspl} but hooks into the entry’s record count.

\begin{verbatim}
\texttt{\textbackslash glspl}[⟨options⟩]{⟨entry-label⟩}{⟨insert⟩}
\end{verbatim}

Like \texttt{\textbackslash glspl} but hooks into the entry’s record count.

\begin{verbatim}
\texttt{\textbackslash GLSplformat}{⟨entry-label⟩}{⟨insert⟩}
\end{verbatim}

Format used by \texttt{\textbackslash GLSp1} if the entry’s record count is more than the given trigger value.

\begin{verbatim}
\texttt{\textbackslash GLSp1format}{⟨entry-label⟩}{⟨insert⟩}
\end{verbatim}

Format used by \texttt{\textbackslash glspl} if the entry’s record count is more than the given trigger value.

\begin{verbatim}
\texttt{\textbackslash glsplformat}{⟨entry-label⟩}{⟨insert⟩}
\end{verbatim}

Format used by \texttt{\textbackslash glspl} if the entry’s record count is more than the given trigger value.

\begin{verbatim}
\texttt{\textbackslash Rho}
\end{verbatim}

Defined with \texttt{\textbackslash providecommand}, this just does \texttt{\textbackslash mathrm{P}}.
Command Summary

S

\seealsoname

\alsoname

(initial: see also) glossaries-extra v1.16+

Used as a cross-reference tag. The default value is \alsoname, if that command has been defined, or “see also”.

\seename

\see

(initial: see (language-sensitive))

Used as a cross-reference tag (provided by glossaries if not already defined).

\setabbreviationstyle\[\langle category\rangle\]{\langle style-name\rangle}

Sets the current abbreviation style to \langle style-name\rangle for the category identified by \langle category\rangle.

If the optional argument is omitted, abbreviation is assumed.

\setacronymstyle\{\langle acronym-style-name\rangle\}

glossaries v4.02+

Sets the style for the base glossaries package’s acronym mechanism. These styles are not compatible with glossaries-extra, which redefines \newacronym to use \newabbreviation. Use:

\setabbreviationstyle[acronym]{\langle abbreviation-style-name\rangle}

with the closest matching abbreviation style instead.

\setentrycounter\[\langle prefix\rangle\]{\langle counter\rangle}

glossaries

Used to set the location counter and prefix required for \glshypernumber.

\setglossarypreamble\[\langle type\rangle\]{\langle text\rangle}

glossaries v3.07+

976
Globally sets the preamble for the glossary identified by \textit{⟨type⟩} to \textit{⟨text⟩}. If \textit{⟨type⟩} is omitted, \texttt{\glstextfont} is assumed.

\begin{verbatim}
\setglossarystyle{⟨style-name⟩} \hfill glossaries v3.08a+
\end{verbatim}

Set the current glossary style to \textit{⟨style-name⟩}. Redefined by glossaries-extra to include \texttt{\glsxtrregglossarystyle}.

\begin{verbatim}
\setupglossaries{⟨options⟩} \hfill glossaries v3.11a+ §2; 9
\end{verbatim}

Change allowed options that are defined by the base glossaries package. Note that some options can only be passed as package options. To change options defined or modified by the glossaries-extra package, use \texttt{\glossariesextrasetup}.

\begin{verbatim}
\setupglsadd{⟨options⟩} \hfill glossaries-extra v1.49+ §5.1.1; 190
\end{verbatim}

Set the default options for \texttt{\glssadd}.

\begin{verbatim}
\setupgllink{⟨options⟩} \hfill glossaries-extra v1.49+ §5.1.1; 190
\end{verbatim}

Set the default \texttt{\gllink} options.

\begin{verbatim}
\subglossentry{⟨level⟩}{⟨entry label⟩}{⟨location list⟩} \hfill glossaries v3.08+
\end{verbatim}

Used to format a child entry. This command should be redefined by the glossary style.

\begin{verbatim}
\SUFFIXOF \hfill §11.5.2; 561
\end{verbatim}

Defined by \texttt{\GlsXtrResourceInitEscSequences} to expand to detokenized \texttt{\SUFFIXOF}.
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\symbolname</td>
<td>Initial: Symbol glossaries (language-sensitive) Expands to the title of the symbol column for headed tabular-like styles.</td>
</tr>
<tr>
<td>\Tau</td>
<td>Defined with \providecommand, this just does \texttt{mathrm{T}}.</td>
</tr>
<tr>
<td>\theentry</td>
<td>The value of the glozaryentry counter.</td>
</tr>
<tr>
<td>\TITLE</td>
<td>Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \TITLE.</td>
</tr>
<tr>
<td>\TRIM</td>
<td>Defined by \GlsXtrResourceInitEscSequences to expand to detokenized \TRIM.</td>
</tr>
<tr>
<td>\u\langle hex\rangle</td>
<td>Recognised by \texttt{bib2gls} within some resource options as identifying the Unicode character given by \langle hex\rangle. Since \u is defined by the \LaTeX kernel as an accent command, you need to protect it from expansion while the options are written to the aux file (\texttt{string\textasciitilde\u\langle hex\rangle}).</td>
</tr>
<tr>
<td>\UC</td>
<td></td>
</tr>
</tbody>
</table>

$\Omega$ 5.2; 561

$\Omega$ 11.5.2; 561
Command Summary

Defined by $\texttt{\textbackslash GlsXtrResourceInitEscSequences}$ to expand to detokenized $\texttt{\textbackslash UC}$.

\[\texttt{\textbackslash Upalpha} \quad \textit{glossaries-extra-bib2gls v1.27+} \quad \texttt{\textsection11.5.8; 599} \]

Defined with $\texttt{\textbackslash providecommand}$ and only if $\texttt{upgreek}$ has been loaded, this just does $\texttt{\textbackslash mathrm \{A\}}$.

\[\texttt{\textbackslash Upbeta} \quad \textit{glossaries-extra-bib2gls v1.27+} \quad \texttt{\textsection11.5.8; 599} \]

Defined with $\texttt{\textbackslash providecommand}$ and only if $\texttt{upgreek}$ has been loaded, this just does $\texttt{\textbackslash mathrm \{B\}}$.

\[\texttt{\textbackslash Upchi} \quad \textit{glossaries-extra-bib2gls v1.27+} \quad \texttt{\textsection11.5.8; 599} \]

Defined with $\texttt{\textbackslash providecommand}$ and only if $\texttt{upgreek}$ has been loaded, this just does $\texttt{\textbackslash mathrm \{X\}}$.

\[\texttt{\textbackslash Upsilonion} \quad \textit{glossaries-extra-bib2gls v1.27+} \quad \texttt{\textsection11.5.8; 599} \]

Defined with $\texttt{\textbackslash providecommand}$ and only if $\texttt{upgreek}$ has been loaded, this just does $\texttt{\textbackslash mathrm \{E\}}$.

\[\texttt{\textbackslash Upeta} \quad \textit{glossaries-extra-bib2gls v1.27+} \quad \texttt{\textsection11.5.8; 599} \]

Defined with $\texttt{\textbackslash providecommand}$ and only if $\texttt{upgreek}$ has been loaded, this just does $\texttt{\textbackslash mathrm \{H\}}$.

\[\texttt{\textbackslash Upiota} \quad \textit{glossaries-extra-bib2gls v1.27+} \quad \texttt{\textsection11.5.8; 599} \]

Defined with $\texttt{\textbackslash providecommand}$ and only if $\texttt{upgreek}$ has been loaded, this just does $\texttt{\textbackslash mathrm \{I\}}$. 

979
### Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Upkappa</td>
<td>Defined with \providecommand and only if upgreek has been loaded, this just does \textup{K}.</td>
</tr>
<tr>
<td>\Upmu</td>
<td>Defined with \providecommand and only if upgreek has been loaded, this just does \textup{M}.</td>
</tr>
<tr>
<td>\Upnu</td>
<td>Defined with \providecommand and only if upgreek has been loaded, this just does \textup{N}.</td>
</tr>
<tr>
<td>\Upomicron</td>
<td>Defined with \providecommand and only if upgreek has been loaded, this just does \textup{O}.</td>
</tr>
<tr>
<td>\upomicron</td>
<td>Defined with \providecommand and only if upgreek has been loaded, this just does \textup{o}.</td>
</tr>
<tr>
<td>\Uprho</td>
<td>Defined with \providecommand and only if upgreek has been loaded, this just does \textup{P}.</td>
</tr>
<tr>
<td>\Uptau</td>
<td>Defined with \providecommand and only if upgreek has been loaded, this just does \textup{p}.</td>
</tr>
</tbody>
</table>

§11.5.8; 599
Command Summary

Defined with \providecommand and only if upgreek has been loaded, this just does \texttt{\textbackslash mathrm \{T\}}.

\texttt{\textbackslash Upzeta} \quad \texttt{\textbackslash providecommand} \quad \texttt{\textbackslash \textit{glossaries-extra-bib2gls v1.27+}} \quad \texttt{$\S$11.5.8; 599}

Defined with \providecommand and only if upgreek has been loaded, this just does \texttt{\textbackslash mathrm \{Z\}}.

\texttt{\textbackslash \textit{W}} \quad \texttt{\textbackslash providecommand} \quad \texttt{\textbackslash \textit{glossaries-extra v1.48+}} \quad \texttt{$\S$7.13; 374}

\texttt{\textbackslash writemultiglossentry\{\textit{\textbackslash options}\}\{\textit{\textbackslash multi-label}\}\{\textit{\textbackslash main-label}\}\{\textit{\textbackslash list}\}} \quad \texttt{\textbackslash \textit{glossaries-extra v1.48+}}

Writes multi-entry information to the \texttt{aux} file.

\texttt{\textbackslash X} \quad \texttt{\textbackslash providecommand} \quad \texttt{\textbackslash \textit{mfirstuc v2.03+}} \quad \texttt{$\S$8.6.5.4; 439}

\texttt{\textbackslash xcapitalisefmtwords\{\textit{\textbackslash text}\}} \quad \texttt{\textbackslash \textit{mfirstuc v2.03+}}

Passes the argument to \texttt{\textbackslash capitalisefmtwords} but with the first token in \texttt{\textit{\textbackslash text}} expanded. The starred version uses the starred version of \texttt{\textbackslash capitalisefmtwords}.

\texttt{\textbackslash xglssetwidest\{\textit{\textbackslash level}\}\{\textit{\textbackslash name}\}} \quad \texttt{\textbackslash \textit{glossaries-extra-stylemods v1.05+}} \quad \texttt{$\S$8.6.5.4; 439}

As \texttt{\textbackslash glssetwidest} but global.

\texttt{\textbackslash xglssupdatewidest\{\textit{\textbackslash level}\}\{\textit{\textbackslash name}\}} \quad \texttt{\textbackslash \textit{glossaries-extra-stylemods v1.23+}} \quad \texttt{$\S$8.6.5.4; 440}

As \texttt{\textbackslash glssupdatewidest} but global.

\texttt{\textbackslash \textit{xGlsXtrIfValueInFieldCsvList\{\textit{\textbackslash entry-label}\}\{\textit{\textbackslash field-label}\}\{\textit{\textbackslash value}\}\{\textit{\textbackslash true}\}\{\textit{\textbackslash false}\}} \quad \texttt{\textbackslash \textit{modifier: *\texttt{\textbackslash \textit{glossaries-extra v1.47+}}}} \quad \texttt{$\S$5.13; 307}
\textbf{Command Summary}

As \texttt{\GlsXtrIfValueInFieldCsvList} but fully expands \texttt{⟨value⟩}.

\begin{verbatim}
\xGlsXtrSetField{⟨entry-label⟩}{⟨field-label⟩}{⟨value⟩}    \
\end{verbatim}

§3.5; 40

As \texttt{\GlsXtrSetField} but expands the value and uses a global assignment.

\begin{verbatim}
\xpglsxtrpostabbrvfootnote
\end{verbatim}

§4.5.2; 160

Expands to the footnote code required for styles like \texttt{short-postfootnote}.

\begin{verbatim}
\xpglsxtrposthyphenlong
\end{verbatim}

152

Used within the post-link hook for the \texttt{short-hyphen-postlong-hyphen} style on first use. This expands the placeholder commands and uses either \texttt{\glsxtrposthyphenlong} or \texttt{\GLSxtrposthyphenlong}.

\begin{verbatim}
\xpglsxtrposthyphenshort
\end{verbatim}

150

Used within the post-link hook for the \texttt{long-hyphen-postshort-hyphen} style on first use. This expands the placeholder commands and uses either \texttt{\glsxtrposthyphenshort} or \texttt{\GLSxtrposthyphenshort}.

\begin{verbatim}
\xpglsxtrposthyphensubsequent
\end{verbatim}

151

Used within the post-link hook for the \texttt{long-hyphen-postshort-hyphen} style on subsequent use. This expands the placeholder commands and uses either \texttt{\glsxtrposthyphensubsequent} or \texttt{\GLSxtrposthyphensubsequent}.

\texttt{Z}

\begin{verbatim}
\Zeta
\end{verbatim}

§11.5.8; 599

Defined with \texttt{\providecommand}, this just does \texttt{\mathrm(Z)}.
Environment Summary

\begin{glstablesubentries} \textit{glossary-table v1.49+}

Encapsulates the child list.

\begin{printunsrtglossarywrap}[\langle options\rangle] \textit{glossaries-extra v1.44+}

Sets up the start and end of the glossary (including beginning and ending the theglossary environment). Use \texttt{\textbackslash printunsrtinnerglossary} within the body for each block of entries.

\begin{theglossary} \textit{glossaries}

Redefined by glossary styles to set up the way the glossary is displayed. For example, to start and end the description environment for the list styles.
Package Option Summary

\usepackage[⟨options⟩]{glossaries-extra-stylemods} (or \usepackage[stylemods=⟨options⟩]{glossaries-extra})

Modifies the glossary styles supplied with the base glossaries package to make them more flexible and to integrate support for features provided by glossaries-extra or bib2gls.

all
Load all predefined styles.

⟨name⟩
Load package glossary-⟨name⟩.

\usepackage[⟨options⟩]{glossaries-extra}

§2; 8

Extension package that loads glossaries, provides additional commands, and modifies some of the base glossaries commands to integrate them with the new commands or to make them more flexible.

abbreviations
Provides a new glossary with the label abbreviations and title given by \abbreviations-name, redefines \glsxtrabbrvtype to abbreviations, redefines \acronymtype to \glsxtrabbrvtype (unless the acronym or acronyms option has been used), and provides \printabbreviations.

accsupp
Loads glossaries-accsupp.

autoseeindex=⟨boolean⟩ default: true; initial: true
Indicates whether or not to enable automatic indexing of see and seealso fields.

autoseeindex=false
Disables automatic indexing of see and seealso fields.

autoseeindex=true
Enables automatic indexing of see and seealso fields.
Package Option Summary

\texttt{bibglsaux=⟨basename⟩} \hspace{1cm} \textit{initial: empty \texttt{glossaries-extra v1.49+}} \hspace{1cm} \texttt{(requires \texttt{bib2gls v3.0+})}

If set, an additional \texttt{aux} file called \texttt{⟨ basename ⟩.aux} will be created in which to store the \texttt{bib2gls} records. This file will be skipped by \LaTeX{} when the main \texttt{aux} file is input but will be read by \texttt{bib2gls}.

\texttt{debug=(value)} \hspace{1cm} \textit{default: true; initial: false \texttt{glossaries v4.24+}}

Provides debugging information. Some options are also available with just the base \texttt{glossaries} package.

\begin{itemize}
\item \texttt{debug=all} \texttt{glossaries-extra v1.21+} \hspace{1cm} \texttt{Implements all debugging options.}
\item \texttt{debug=false} \texttt{glossaries v4.24+} \hspace{1cm} \texttt{Disable debugging actions.}
\item \texttt{debug=showaccsupp} \texttt{glossaries v4.45+} \hspace{1cm} \texttt{Implements debug=true and also shows accessibility information in the document.}
\item \texttt{debug=showtargets} \texttt{glossaries v4.24+} \hspace{1cm} \texttt{Implements debug=true and also shows target markers in the document.}
\item \texttt{debug=showwrgloss} \texttt{glossaries-extra v1.21+} \hspace{1cm} \texttt{Implements debug=true and also shows a marker in the document whenever the entry is indexed and (if used with \texttt{indexcounter}) will mark where the wrglossary counter is incremented.}
\item \texttt{debug=true} \texttt{glossaries v4.24+} \hspace{1cm} \texttt{Writes \texttt{wrglossary(⟨type⟩)(⟨indexing info⟩)} to the log file if there is an attempt to index an entry before the associated indexing file has been opened (\texttt{makeindex} and \texttt{xindy} only). With \texttt{glossaries-extra}, this setting will also display the label of any undefined entries that are referenced in the document.}
\end{itemize}

\texttt{docdef=(value)} \hspace{1cm} \textit{default: true; initial: false \texttt{glossaries v4.24+}} \hspace{1cm} \texttt{§2.4; 16}

Determines whether or not \texttt{\newglossaryentry} is permitted in the document environment.

\begin{itemize}
\item \texttt{docdef=atom} \hspace{1cm} \texttt{As restricted but creates the \texttt{glsdefs} file for atom’s autocomplete support.}
\item \texttt{docdef=false} \hspace{1cm} \texttt{Don’t allow \texttt{\newglossaryentry} in the document environment.}
\item \texttt{docdef=restricted} \hspace{1cm} \texttt{Allow \texttt{\newglossaryentry} in the document environment, but only before any glossaries.}
\item \texttt{docdef=true}
\end{itemize}
Package Option Summary

Allow \newglossaryentry in the document environment if the base glossaries package would allow it.

\textbf{equations} \langle \text{boolean} \rangle  \hspace{1cm} \text{default: true; initial: false} \hspace{1cm} \S 2.4; 26

Automatically switch the location counter to equation when inside a numbered equation environment.

\textbf{floats} \langle \text{boolean} \rangle  \hspace{1cm} \text{default: true; initial: false} \hspace{1cm} \S 2.4; 27

Automatically switch the location counter to the corresponding counter when inside a floating environment.

\textbf{indexcounter} \hspace{1cm} \S 2.4; 27

Defines the index counter \texttt{wrglossary} and implements \texttt{counter=wrglossary}.

\textbf{indexcrossrefs} \langle \text{boolean} \rangle  \hspace{1cm} \text{default: true; initial: varies} \hspace{1cm} \S 2.4; 20

Indicates whether or not to enable automatic indexing of cross-referenced entries.

\textbf{indexcrossrefs=false} \hspace{1cm} \S 2.4; 20

Disables automatic indexing of cross-referenced entries.

\textbf{indexcrossrefs=true} \hspace{1cm} \S 2.4; 20

Enables automatic indexing of cross-referenced entries.

\textbf{nomissingglstext} \langle \text{boolean} \rangle  \hspace{1cm} \text{default: true; initial: false} \hspace{1cm} \S 2.1; 9

Determines whether or not to display warning text if the external glossary file hasn’t been generated due to an incomplete build.

\textbf{postdot} \hspace{1cm} \S 2.2; 12

A shortcut for \texttt{nopostdot=false}.

\textbf{postpunc} \langle \text{value} \rangle  \hspace{1cm} \S 2.2; 12

An alternative to \texttt{postdot}, this can be used to insert a different punctuation character after the description.

\textbf{postpunc=comma} \hspace{1cm} \S 2.2; 12

Inserts a comma after the description.

\textbf{postpunc=dot} \hspace{1cm} \S 2.2; 12

Equivalent to \texttt{postdot} or \texttt{nopostdot=false}.

\textbf{postpunc=none} \hspace{1cm} \S 2.2; 12

Switches off automatic post-description punctuation insertion.

\textbf{postpunc=\langle punctuation \rangle} \hspace{1cm} \S 2.2; 12

Inserts \texttt{\langle punctuation \rangle} after the description.

\textbf{prefix} \hspace{1cm} \S 2.3; 14
Package Option Summary

Loads glossaries–prefix.

**record=(value)**

*default: only; initial: off*

Indicates whether or not bib2gls is being used (in which case entry indexing is performed by adding bib2gls records in the aux file).

*record=alsoindex*

*alias: hybrid*

Performs a mixture of bib2gls records in the aux file (to select entries from a bib file) and makeindex/xindy indexing in their associated files. The actual sorting and collation is performed by the indexing application, so `sort=none` and `save-locations=false` should be used in \GlsXtrLoadResources (because it’s redundant to make bib2gls sort and collate as well). This setting should be used with \makeglossaries before \GlsXtrLoadResources and glossaries should be displayed with \printglossary (or \printglossaries). There’s little point in using this setting unless you have a custom xindy module that you can’t convert to an equivalent set of bib2gls options.

*record=nameref*

Entry indexing is performed by adding bib2gls nameref records in the aux file. Glossaries should be displayed with the “unsrt” family of commands.

*record=off*

Entry indexing is performed as per the base glossaries package, using either \makeglossaries or \makenoidxglossaries.

*record=only*

Entry indexing is performed by adding bib2gls records in the aux file. Glossaries should be displayed with the “unsrt” family of commands.

**shortcuts=(value)**

*initial: none*

Defines various shortcut commands (boolean only with just the base glossaries package).

*shortcuts=abbr*

*alias: abbreviations*

Implements the abbreviation shortcuts.

*shortcuts=abbreviations*

Implements shortcuts=abbreviations and shortcuts=other.

*shortcuts=ac*

Implements the acronym shortcuts that are compatible with \newabbreviation.

*shortcuts=acother*
Implement short cuts=ac and shortcuts=other.

**shortcuts=acro**

Implement shortcuts=acro alias: acronyms 19

**shortcuts=acronyms**

Implements the acronym shortcuts. Don’t use this option unless you have reverted \newacronym back to the base glossaries package’s acronym mechanism.

**shortcuts=all**

Implements shortcuts=ac, shortcuts=abbreviations, shortcuts=other.

**shortcuts=false**

alias: all 20

**shortcuts=None**

Don’t define any shortcut commands.

**shortcuts=other**

Implements the shortcuts \newentry, \newsym and \newnum.

**shortcuts=true**

alias: all 19

**showtargets=⟨value⟩**

Implements debug=showtargets.

**showtargets=annoteleft**

Markers are placed on either side of the link/target with the annotation on the left in all modes.

**showtargets=annoteright**

Markers are placed on either side of the link/target with the annotation on the right in all modes.

**showtargets=innerleft**

A marker and annotation are placed to the left of the link/target in all modes.

**showtargets=innerright**

A marker and annotation are placed to the left of the link/target in all modes.

**showtargets=left**

A marker is placed to the left of the link/target and a marginal note is used in outer mode.

**showtargets=right**

A marker is placed to the right of the link/target and a marginal note is used in outer mode.

**stylemods=⟨value⟩**

default: default 13 §2.2; 13
Loads glossaries-extra-stylemods with the given options. If \texttt{stylemods=default} then no options are passed to glossaries-extra-stylemods.

- \texttt{stylemods=all}
  Load glossaries-extra-stylemods with all predefined styles.

- \texttt{stylemods=default}
  Load glossaries-extra-stylemods with no options (patches any predefined styles that have been loaded).

- \texttt{stylemods=⟨list⟩}
  Load glossaries-extra-stylemods with all the listed styles.

- \texttt{undefaction=⟨value⟩}
  \textit{initial: error}
  Indicates whether to trigger an error or warning if an unknown entry label is referenced.

- \texttt{undefaction=error}
  Trigger an error if an unknown entry label is referenced.

- \texttt{undefaction=warn}
  Trigger a warning if an unknown entry label is referenced.

\begin{verbatim}
\usepackage[⟨options⟩]{glossaries} (automatically loaded with \usepackage{glossaries-extra})
\end{verbatim}

Base package. When loaded implicitly with glossaries-extra, any relevant options will be passed to the base glossaries package.

- \texttt{acronym=⟨boolean⟩}
  \textit{default: true; initial: false}
  If true, provides a new glossary with the label acronym and title given by \texttt{\acronymname}, redefines \texttt{\acronymtype} to acronym, and provides \texttt{\printacronyms}.

- \texttt{acronymlists=⟨label-list⟩}
  Identifies the glossaries that contain acronyms (defined with the base glossaries packages acronym mechanism).

- \texttt{acronyms}
  Provides a new glossary with the label acronym, redefines \texttt{\acronymtype} to acronym, and provides \texttt{\printacronyms}.

- \texttt{automake=⟨value⟩}
  \textit{default: true; initial: false}
  Indicates whether or not to attempt to use \TeX{}’s shell escape to run an indexing application.


**Package Option Summary**

`counter=(counter-name)  \hspace{1cm} initial: page \equiv`
Sets the default location counter.

`debug=(value)  \hspace{1cm} initial: false \equiv glossaries v4.24+`
(modified by glossaries-extra’s `debug` option)
Adds markers to the document for debugging purposes.

`entrycounter` \equiv
Enables the entry counter for top-level entries.

`hyperfirst=(boolean)  \hspace{1cm} default: true; initial: true \equiv`
Indicates whether or not to use hyperlinks on first use (if hyperlinks are supported).

`index` \equiv §2.1; 11
Provides a new glossary with the label `index` and the title \indexname, and provides \print-index and \newterm.

`indexonlyfirst=(boolean)  \hspace{1cm} default: true; initial: false \equiv`
Indicates whether or not to only index the first use.

`makeindex` \equiv
Indicates that the indexing should be performed by `makeindex` (default).

`mfirstuc` \equiv glossaries v4.50+
Implements the expanded and unexpanded options provided by `mfirstuc`.

`nogroupskip=(boolean)  \hspace{1cm} default: true; initial: false \equiv`
If true, suppress the gap between letter groups in the glossaries by default.

`nolist` \equiv
Don’t load glossary-list, which is normally loaded automatically. Note that if glossaries is loaded before glossaries-extra, then this option should be passed directly to glossaries not glossaries-extra otherwise it will be too late to implement.

`nolong` \equiv
Don’t load glossary-long, which is normally loaded automatically. Note that if glossaries is loaded before glossaries-extra, then this option should be passed directly to glossaries not glossaries-extra otherwise it will be too late to implement.

`nomain` \equiv
Prevents the definition of the main glossary. You will need to define another glossary to use instead. For example, with the `acronyms` package option.

`nonumberlist` \equiv
Set no location lists as the default for all glossaries. May be overridden for individual glossaries with `nonumberlist=true`.

990
nopostdot=⟨boolean⟩

If true, suppresses the automatic insertion of a full stop after each entry’s description in the glossary (for styles that support this). The default is nopostdot=true for glossaries-extra and nopostdot=false for just glossaries.

noredefwarn

Suppresses a warning if theglossary or \printglossary have already been defined (which indicates that the document class or another package also provides a mechanism for creating a glossary that could potentially conflict with glossaries). This option is automatically implemented with glossaries-extra.

nostyles

Don’t load the default set of predefined styles. Note that if glossaries is loaded before glossaries-extra, then this option should be passed directly to glossaries not glossaries-extra otherwise it will be too late to implement.

nosuper

Don’t load glossary-super, which is normally loaded automatically. Note that if glossaries is loaded before glossaries-extra, then this option should be passed directly to glossaries not glossaries-extra otherwise it will be too late to implement.

notranslate

Equivalent to translate=false.

notree

Don’t load glossary-tree, which is normally loaded automatically. Note that if glossaries is loaded before glossaries-extra, then this option should be passed directly to glossaries not glossaries-extra otherwise it will be too late to implement.

nowarn

Suppresses warnings.

numberedsection=⟨value⟩

Indicates whether or not glossary section headers will be numbered and also if they should automatically be labelled.

numbers

Provides a new glossary with the label numbers and the title \glsnumbersgrouplname, and provides \printnumbers. With glossaries-extra, this additionally defines \glsextrnewnumber.

sanitizesort=⟨boolean⟩

Indicates whether the default sort value should be sanitized (only applicable with sort=standard).

savenumberlist=⟨boolean⟩

If true, patches the location list encapsulator to save the location list. With bib2gls, use the save-locations resource option.
Package Option Summary

section=(value)  
Indicates which section heading command to use for the glossary. The value may be one of the standard sectioning command's control sequence name (without the leading backslash), such as chapter or section.

seenoindex=(value)  
Indicates what to do if the see key is used before the associated indexing files have been opened by \makeglossaries.

  seenoindex=error
  Triggers an error if the see key is used before \makeglossaries.

  seenoindex=ignore
  Does nothing if the see key is used before \makeglossaries.

  seenoindex=warn
  Triggers a warning if the see key is used before \makeglossaries.

sort=(value)  
Indicates how the sort key should automatically be assigned if not explicitly provided (for \makeglossaries and \makenoidxglossaries only).

  sort=clear glossaries v4.50+
  Sets the sort key to an empty value. Use this option if no indexing is required for a slightly faster build.

  sort=def
  Use the (zero-padded) order of definition as the default for the sort key.

  sort=none glossaries v4.30+
  Don't process the sort key. Use this option if no indexing is required for a slightly faster build.

  sort=standard
  Use the value of the name key as the default for the sort key and implement the \glsprestandardsort hook.

  sort=use
  Use the (zero-padded) order of use as the default for the sort key.

style=(style-name)  
Sets the default glossary style to (style-name).

subentrycounter  
Enables the entry counter for level 1 entries.
**Package Option Summary**

`symbols`
Provides a new glossary with the label `symbols` and the title `\glssymbolsgroupname`, and provides `\printsymbols`. With `glossaries-extra`, this additionally defines `\gxtrnewsymbol`.

`toc=(boolean)`
If true, each glossary will be automatically added to the table of contents. The default is `toc=false` with `glossaries` and `toc=true` with `glossaries-extra`.

`translate=(value)`
Indicates how multilingual support should be provided, if applicable.

- `translate=babel`
  Uses `babel`’s language hooks to implement multilingual support (default for `glossaries-extra` if `babel` has been detected).

- `translate=false`
  Don’t implement multilingual support (default if no language package has been detected).

- `translate=true`
  Uses `translator`’s language hooks to implement multilingual support (default for `glossaries` if a language package has been detected).

`writeglslabelnames`
Creates a file called `\jobname.glslabels` that contains all defined entry labels and names (for the benefit of autocompletion tools).

`writeglslabels`
Creates a file called `\jobname.glslabels` that contains all defined entry labels (for the benefit of autocompletion tools).

`xindy={⟨options⟩}`
Indicates that the indexing should be performed by `xindy`.
# Index

**Symbols**

<table>
<thead>
<tr>
<th></th>
<th>@</th>
</tr>
</thead>
<tbody>
<tr>
<td>\</td>
<td>§11.5.2; 494, 561</td>
</tr>
<tr>
<td>&amp;</td>
<td>see tabulation (&amp;)</td>
</tr>
<tr>
<td>\</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>?</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>?? (unknown entry marker)</td>
<td>15, 28, 37, 38, 536, 537, 732, 923, 924</td>
</tr>
<tr>
<td>/</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>.</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>-</td>
<td>see full stop (.)</td>
</tr>
<tr>
<td>\</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>-</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>- (non-breaking space)</td>
<td>see non-breaking space (-)</td>
</tr>
<tr>
<td>&quot;</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td></td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>) (range end)</td>
<td>261, 262, 733</td>
</tr>
<tr>
<td>( (range start)</td>
<td>261, 262, 797</td>
</tr>
<tr>
<td>\</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>[</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>]</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>$</td>
<td>§11.5.2; 560, 561, 727</td>
</tr>
<tr>
<td>*</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>*(modifier)</td>
<td>see \GlsXtrSetStarModifier</td>
</tr>
<tr>
<td>&amp;</td>
<td>§11.5.2; 304, 561</td>
</tr>
<tr>
<td>#</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>#</td>
<td>560, 757</td>
</tr>
<tr>
<td>-</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>+</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>+(modifier)</td>
<td>see \GlsXtrSetPlusModifier</td>
</tr>
<tr>
<td>&lt;</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>&lt;=</td>
<td>§11.5.2; 561</td>
</tr>
<tr>
<td>\</td>
<td>§11.5.2; 561</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>@</th>
</tr>
</thead>
<tbody>
<tr>
<td>@currentHref</td>
<td>24, 25, 581, 583, 899</td>
</tr>
<tr>
<td>@currentLabel</td>
<td>24, 25</td>
</tr>
<tr>
<td>@currentLabelname</td>
<td>581</td>
</tr>
<tr>
<td>@endfortrue</td>
<td>305</td>
</tr>
<tr>
<td>@firstofone</td>
<td>241, 299, 586</td>
</tr>
<tr>
<td>@firstoftwo</td>
<td>52, 214</td>
</tr>
<tr>
<td>@for</td>
<td>305, 374</td>
</tr>
<tr>
<td>@gobble</td>
<td>415, 429</td>
</tr>
<tr>
<td>\glsxtr@doglossary</td>
<td>§8.4.3; 404, 406, 407, 680</td>
</tr>
<tr>
<td>\glsxtr@multientry</td>
<td>§7.13; 374, 680</td>
</tr>
<tr>
<td>\glsxtr@org@starttoc</td>
<td>§5.3.3; 227, 680</td>
</tr>
<tr>
<td>\glsxtr@org@markboth</td>
<td>§5.3.3; 227, 680</td>
</tr>
<tr>
<td>\glsxtr@org@markright</td>
<td>§5.3.3; 227, 680</td>
</tr>
<tr>
<td>\glsxtrinmark</td>
<td>§5.3.3; 227, 423, 680</td>
</tr>
<tr>
<td>\glsxtrnotinmark</td>
<td>§5.3.3; 227, 423, 681</td>
</tr>
<tr>
<td>@secondoftwo</td>
<td>214</td>
</tr>
<tr>
<td>@starttoc</td>
<td>201, 204, 214, 227, 423, 680, 861, 877, 901</td>
</tr>
</tbody>
</table>

**A**

| \AB | §4.3.2; Table 4.1; 681 |
| \Ab | §4.3.2; Table 4.1; 681 |
| \ab | §4.3.2; Table 4.1; 552, 681 |

abbreviation styles | 654 |

footnote-desc | see short-footnote-desc |
footnote-em | see short-em-footnote |
footnote-sc | see short-sc-footnote |
footnote-sm | see short-sm-footnote |
footnote | see short-footnote |
long-desc-em | see long-noshort-em-desc |
long-desc-sc | see long-noshort-sc-desc |

994
Index

long-desc-sm .......................... see long-noshort-sm-desc
long-desc ............. see long-noshort-desc
long-em ............ see long-noshort-em-desc-noreg
               131, 655
long-em-noshort-em-desc .. 80, 131, 655
long-em-noshort-em-noreg 64, 131, 655
long-em-noshort-em ... 64, 79, 131, 655
long-em-short-em-desc ... 82, 87, 655
long-em-short-em ... 63, 81, 86, 132, 656
long-em .................. see long-noshort-em
long-hyphen-noshort-desc-noreg 107, 149, 656, 873
long-hyphen-noshort-noreg 64, 107, 149, 656, 873
long-hyphen-postshort-hyphen-desc 106, 656
long-hyphen-short-hyphen-desc 105, 106, 656
long-hyphen-short-hyphen 103–105, 148, 149, 521, 656, 657, 873, 874
long-noshort-desc-noreg 107, 131, 657
long-noshort-desc ... Table 4.2; 73, 74, 76–78, 80, 131, 147, 168, 655, 655, 656, 657, 658, 874
long-noshort-em-desc 73, 78, 654, 657
long-noshort-em ........... 78, 656, 657
long-noshort-noreg ........... 107, 131, 657
long-noshort-sc-desc 73, 76, 654, 657
long-noshort-sc ............. 63, 75, 658, 659
long-noshort-sm-desc 73, 77, 655, 658
long-noshort-sm .......... 76, 658, 661
long-noshort Table 4.2; 47, 74–76, 78, 79, 131, 147, 655, 657, 658, 661, 874
long-only-short-only-desc 112, 113, 155, 156, 658, 890
long-only-short-only 41, 111, 112, 154, 658, 659, 890
long-only-short-sc-only-desc 113, 659, 902
long-only-short-sc-only ........ 112, 659, 707, 902
long-postshort-sc-user-desc ... 92, 138, 659, 877
long-postshort-sc-user ... 91, 92, 138, 337, 659, 877, 903
long-postshort-user-desc ... 90, 659
long-postshort-user ... 90, 91, 142, 239, 659, 897
long-sc .................. see long-noshort-sc
long-short-desc Table 4.2; 82–84, 86, 87, 133, 134, 655, 659, 660, 875
long-short-em-desc ............ 82, 86, 660
long-short-em ............ 63, 81, 85, 660
long-short-sc-desc Table 4.2; 82, 83, 660
long-short-sc ... Table 4.2; 41, 42, 44, 81, 82, 171, 172, 660
long-short-sm-desc ... Table 4.2; 82, 84, 660
long-short-sm ... Table 4.2; 81, 84, 660
long-short-user-desc ........... 89, 91, 138, 661, 877
long-short-user 64, 88–90, 135, 139–142, 659, 661, 814, 928, 930–932
long-short Table 4.2; 3, 42, 44, 50, 60, 67, 73, 81, 82, 84–86, 88, 103, 132, 133, 165, 171, 241, 654, 656, 657, 660, 661, 876
long-sm .................. see long-noshort-sm
long .................. see long-noshort
nolong-short-em ........... 67, 73, 661
nolong-short-noreg .......... 130, 661
nolong-short-sc ............. 67, 69, 662
nolong-short-sm .......... 67, 71, 662
nolong-short 67, 69, 71, 73, 130, 661, 662
postfootnote-desc ............. see short-postfootnote-desc
postfootnote-em ................ see short-em-postfootnote
postfootnote-sc ................ see short-sc-postfootnote
postfootnote-sm ................ see short-sm-postfootnote
postfootnote ... see short-postfootnote
short-desc ............. see short-nolong-desc

995
Index

short-em-desc see short-em-nolong-desc
short-em-footnote-desc 64, 127, 663
short-em-footnote 64, 114, 126, 654, 663
short-em-long-desc 94, 97, 663
short-em-long-em 94, 99, 663
short-em-long 93, 98, 663
short-em-nolong-desc 66, 72, 73, 663, 664
short-em-nolong 65, 71, 72, 664
short-em-postfootnote-desc 129, 664
short-em-postfootnote 128, 662, 664
short-em see short-em-nolong
short-footnote Table 4.2; 115–117, 119, 123, 127, 144, 145, 654, 663, 664, 667, 669, 844
short-footnote Table 4.2; 50, 64, 114–116, 118, 122, 126, 654, 663, 664, 667, 669, 820, 844
short-hyphen-long-hyphen-desc 110, 665
short-hyphen-long-hyphen 108–110, 149, 152, 665, 857, 910
short-hyphen-postlong-hyphen-desc 110, 665
short-hyphen-postlong-hyphen 109, 152, 153, 665, 894, 909, 982
short-long-desc Table 4.2; 64, 94–97, 99, 134, 663, 665, 668, 690, 910
short-long-user-desc 100, 103, 139, 666, 911
short-long-user 100, 140, 141, 143, 666, 929, 931, 932
short-long Table 4.2; 64, 65, 93, 94, 96–98, 100, 101, 108, 114, 134, 663–665, 666, 667–669, 911
short-nolong-desc-noreg 130, 666
short-nolong-desc 66–68, 70, 72, 130, 147, 663, 664, 666, 668, 669, 908
short-nolong-noreg 130, 666
short-postfootnote-desc 117, 118, 121, 125, 129, 662, 664, 667, 668, 670
short-postlong-user-desc 102, 667
short-postlong-user 54, 101, 102, 144, 667, 897
short-sc-desc see short-sc-nolong-desc
short-sc-footnote-desc Table 4.2; 119, 667
short-sc-footnote Table 4.2; 114, 118, 654, 667
short-sc-long-desc Table 4.2; 94, 95, 668
short-sc-long Table 4.2; 93, 94, 668
short-sc-nolong-desc 66, 68, 69, 667, 668
short-sc-nolong 65, 67, 68, 668
short-sc-postfootnote-desc 121, 668
short-sc-postfootnote 64, 120, 662, 668
short-sc see short-sc-nolong
short-sm-desc see short-sm-nolong-desc
short-sm-footnote-desc Table 4.2; 123, 669
short-sm-footnote Table 4.2; 114, 122, 654, 669
short-sm-long-desc Table 4.2; 94, 96, 669
short-sm-long Table 4.2; 93, 96, 669, 914
short-sm-nolong-desc 66, 70, 71, 669
short-sm-nolong 65, 69, 70, 669, 670
short-sm-postfootnote-desc 125, 670
short-sm-postfootnote 124, 662, 670
short-sm see short-sm-nolong
\abbreviationsname §2.1; 9, 10, 618, 681, 984
\abbrvpluralsuffix 36, 43, 44, 170, 171, 501, 681, 686
\ABP §4.3.2; Table 4.1; 682
\Abp §4.3.2; Table 4.1; 682
\abp §4.3.2; Table 4.1; 682
\AC §4.3.2; Table 4.1; 682
Index

\ac §4.3.2; Table 4.1; 18, 19, 43, 311, 552, 682 short-long-desc ............. Table 4.2
accsupp package ............. 717 short-long .................. Table 4.2
\ACF §4.3.2; Table 4.1; 682 sm-short-long-desc ................. Table 4.2
\Acf §4.3.2; Table 4.1; 683 sm-short-long ................. Table 4.2
\acf §4.3.2; Table 4.1; 683 \acronymfont .................. 230, 686
\ACFP §4.3.2; Table 4.1; 683 \acronymname 9, 618, 681, 686, 989
\Acfp §4.3.2; Table 4.1; 683 \acronymtype §4.1.4; 10, 11, 44, 686, 968,
\acfp §4.3.2; Table 4.1; 683 970, 984, 989
\ACL §4.3.2; Table 4.1; 683 \acrpluralsuffix 36, 686
\Acl §4.3.2; Table 4.1; 683 \Acrshort ☐ ............. 687
\acl §4.3.2; Table 4.1; 19, 684 \acrshort ☐ 18, 19, 50, 687
\ACLp §4.3.2; Table 4.1; 684 \Acrshortpl ☐ ......... 687, 688
\Aczp §4.3.2; Table 4.1; 684 \acrshortpl ☐ ......... 687, 688
\aclp §4.3.2; Table 4.1; 684 \acs §4.3.2; Table 4.1; 687
\ACP §4.3.2; Table 4.1; 684 \acs §4.3.2; Table 4.1; 687
\Acp §4.3.2; Table 4.1; 684 \acs §4.3.2; Table 4.1; 19, 687
\acp §4.3.2; Table 4.1; 685 \ACSP §4.3.2; Table 4.1; 688
\Acrfull ☐ ..... 683, 685 \Acsp §4.3.2; Table 4.1; 688
\acrfull ☐ ..... 683, 685 \acsp §4.3.2; Table 4.1; 688
\Acrfullpl ☐ ..... 683, 685 \actualchar ............. 602
\acrfullpl ☐ ..... 683, 685 \AF §4.3.2; Table 4.1; 688
\Acrlong ☐ ..... 684, 685 \Af §4.3.2; Table 4.1; 688
\acrlong ☐ 18, 19, 684, 685, 686 \af §4.3.2; Table 4.1; 688
\Acrlongpl ☐ ..... 684, 686 \AFP §4.3.2; Table 4.1; 688
\acrlongpl ☐ ..... 684, 686 \AFP §4.3.2; Table 4.1; 689
acronym (base) style
  dua-desc ............. Table 4.2 \AL §4.3.2; Table 4.1; 689
  dua ................... Table 4.2 \A1 §4.3.2; Table 4.1; 689
  footnote-desc ........ Table 4.2 \a1 §4.3.2; Table 4.1; 689
  footnote-sc-desc .......... Table 4.2 align environment .......... 26
  footnote-sc ............. Table 4.2 all caps 200, 254, 727, 886, 887, 947, 948
  footnote-sm-desc ........ Table 4.2 \ALP §4.3.2; Table 4.1; 689
  footnote-sm ............. Table 4.2 \A1p §4.3.2; Table 4.1; 689
  footnote ................ Table 4.2 \alp §4.3.2; Table 4.1; 689
  long-sc-short-desc ........ Table 4.2 \Alph §11.5.8; 567, 578, 579, 599, 690
  long-sc ................ Table 4.2 \alph §265, 266
  long-short ................ Table 4.2 \alpha §265
  long-short-desc ........ Table 4.2 \Alpha §11.5.8; 567, 578, 579, 599, 690
  long-short ............. 229 \alpha §567, 577–579
  long-sm-short-desc ........ Table 4.2 \alsoname 690, 976
  long-sm-short ........... Table 4.2 \langle alt-mod \rangle (modifier) see
  long-sp-short-desc ........ Table 4.2 \GlsXtrSetAltModifier
  long-sp-short ........... Table 4.2 amsmath package 27
  sc-short-long-desc ........ Table 4.2 \andname 304, 690
  sc-short-long ........... Table 4.2 \appto 164, 396, 404, 406, 500
Index

\apptoglossarypreamble  §8.2; 381, 690
\arabic  265, 266
arara  536
array package  452, 482
\AS  §4.3.2; Table 4.1; 690
\As  §4.3.2; Table 4.1; 690
\as  §4.3.2; Table 4.1; 690
ASCII  389, 546, 625, 889, 907
\ASP  §4.3.2; Table 4.1; 691
\Asp  §4.3.2; Table 4.1; 691
\asp  §4.3.2; Table 4.1; 691
B
babel package  3, 4, 9, 17, 517, 618, 681, 690, 993
\backmatter  188
beamer class  289
\Beta  §11.5.8; 567, 578, 599, 691
\beta  567, 577, 579
bib2gls  24, 26, 34, 276, 394, 399, 402, 431, 448, 454, 479, 488, 544, 564, 633
--group  25
--merge-nameref-on  586
--no-group  650
--record-count-rule  547
--record-count-unit  547, 553, 557, 641
--record-count  268, 547, 553, 640, 641
\bibgls@input  691
\bibglsdualprefixlabel  §11.5.7; 597, 691
\bibglsnewdualindexsymbolsecondary  691
\bibglsprimaryprefixlabel  §11.5.7; 597, 692
\bibglssetlastgrouptitle  563, 692
\bibglssetlocationrecordcount  §11.4.2; 557, 692, see also --record-count-unit
\bibglstotalrecordcount  §11.4.2; 556, 692, see also --record-count
\bibglstertiaryprefixlabel  §11.5.7; 597, 692
booktabs package  482, 490, 673
\bottomrule  470
C
\capitalisefmtwords  200, 525, 692, 727, 981
\capitalisewords  200, 294, 376, 525, 693, 727, 842
\caption  369, 524, 593, 941, see also uppercase, lowercase, title case, sentence case & all caps
\CAT  §11.5.2; 561, 693
categories
abbreviation §10.1; 41, 42, 58–60, 159, 239, 241, 323, 428, 518, 654, 892, 958, 976
acronym  §10.1; 5, 42, 46, 58–60, 181, 237, 239, 241, 427, 518, 520, 654, 892, 958
genereal  §10.1; 239, 246, 254, 314, 315, 319, 427, 484, 518, 520, 528, 530, 892
index  §10.1; 12, 518, 893
number  §10.1; 11, 519, 893
symbol  §10.1; 10, 408, 519, 893
term  427
category attributes  248, 634, 636, 836
accessaposplural  §9.1; 501
accessinsertdots  §9.1; 501
accessnoshortplural  §9.1; 501
aposplural  §10.2.1.1; 43, 162, 501, 522, 836
combinedfirstsep  §7.8; 347, 357
combinedfirstsepfirst  §7.8; 347, 357
combinedsep  §7.8; 346, 357
combinedsepfirst  §7.8; 347, 357
discardperiod  §10.2.1.1; 248, 333, 520, 522, 836

998
Index

dualindex §10.2.1.4; 527, 600, 602, 612
encapinnerfmt ...................... 244
encapnocaseinnerfmt .............. 244
entrycount ................................ 311, 312, 314–316, 526
externallocation §10.2.1.4; 195, 196, 528, 581, 917
firstshortaccess §9.1; 168, 169, 502, 632, 633
glossdesc §10.2.1.3; 197, 200, 235, 424, 524, 526
glossname §10.2.1.3; 197, 424, 425, 464, 526

glossnamefont §10.2.1.3; 197, 424, 425, 464, 526
headuc §10.2.1.4; 204, 216–218, 526, 852–856, 860
hyperoutside §10.2.1.2; 190, 524
indexname §10.2.1.3; 425, 526, 600, 612
linkcount §10.2.1.4; 320, 323, 526
markshortwords §10.2.1.1; 43, 63, 103, 148, 162, 167, 168, 174, 177, 522, 657, 665, 749, 818, 933, 934
markwords §10.2.1.1; 43, 63, 103, 148, 162, 167, 176, 521, 522, 656, 657, 665, 749, 819, 933, 934
multioptions §7.8; 335, 356, 357
nameshortaccess §9.1; 168, 169, 501, 631
nohyper §10.2.1.2; 257, 524, 530
nohyperfirst §10.2.1.2; 115, 116, 187, 189, 257, 523, 665, 832
nohypernext §10.2.1.2; 187, 257, 523
noshortplural §10.2.1.1; 44, 162, 501, 522, 523, 681, 788
pluraldiscardperiod §10.2.1.1; 520
recordcount 549–551, 862, 907

regular §10.2.1.1; 519
retainfirstuseperiod §10.2.1.1; 248, 333, 520, 836
tagging §10.2.1.1; 58, 59, 523
targetcategory §10.2.1.4; 528
targetname §10.2.1.4; 527, 528
targeturl §10.2.1.4; 36, 258, 527, 614
textformat §10.2.1.2; 191, 239, 240, 524
textshortaccess §9.1; 168, 169, 502, 634, 637
unitcount §10.2.1.2; 316
wrgloss §10.2.1.2; 185, 189, 194, 267, 524
category post-description hook 10–12, 427, 428, 434, 625, 629, 893
category post-link hook 236, 238, 246–249, 251, 252, 428, 589, 625, 629, 896
category post-name hook §6.1; 589, 625
\cGLS §6.1; Table 4.1; 314, 681, 682, 693, 694, see also \glsenableentrycount & \cGLSformat
\cGls §6.1; 313, 681, 682, 693, 694, see also \glsenableentrycount & \cGlsformat
\cgl §6.1; Table 4.1; 56, 183, 311, 312, 316, 526, 549, 552, 681, 682, 693, 694, see also \glsenableentrycount & \cglformat
\cGLSformat §6.1; 314, 694
\cGlsformat §6.1; 313, 694
\cglformat §6.1; 312–314, 549, 550, 694
\cGLSpl §6.1; Table 4.1; 314, 682, 684, 694, 695, see also \glsenableentrycount & \cGLSplformat
\cGlspl §6.1; 313, 682, 684, 694, 695, see also \glsenableentrycount & \cGlsplformat
\cglspl §6.1; 313, 682, 684, 694, 695, see also \glsenableentrycount & \cglsplformat

999
Index

\cGlsplformat \hspace{1cm} §6.1; 314, 695
\cglspformat \hspace{1cm} §6.1; 313, 314, 695
\chapter \hspace{1cm} 436, 704
chapter counter \hspace{1cm} 316, 554, 555, 558, 581
\char \hspace{1cm} 388
\Chi \hspace{1cm} §11.5.8; 599, 695
CLI \hspace{1cm} 625, 627
convertgls2bib \hspace{1cm} 276, 533
--index-conversion \hspace{1cm} 276
\CS \hspace{1cm} §11.5.2; 561, 695
\cs \hspace{1cm} §11.5.2; 561, 695
\csdef \hspace{1cm} 38, 39, 869
\csGlsXtrLetField \hspace{1cm} §3.5; 39, 40, 695
\cslet \hspace{1cm} 40, 695, 869
\csletcs \hspace{1cm} 40, 695
\currentglossary \hspace{1cm} 403, 419, 437, 695
\CurrentTrackedDialect \hspace{1cm} 619
\CurrentTrackedLanguage \hspace{1cm} 619
\CustomAbbreviationFields \hspace{1cm} §4.5.3.1; 165, 167, 168, 696

D

datatool-base package \hspace{1cm} 305, 306, 699, 841, 859, 862
datatool package \hspace{1cm} 307, 532, 699, 758, 841
\defglsentryfmt \hspace{1cm} 30, 32, 699, 758, 841
\delimN \hspace{1cm} 696, 757, 786
\delimR \hspace{1cm} 696, 757, 786
\Delta \hspace{1cm} 578
description environment \hspace{1cm} 672, 983
descriptionname \hspace{1cm} 451, 490, 499, 696
\dGLS \hspace{1cm} §11.5.7; 591, 696, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\dGls \hspace{1cm} §11.5.7; 591, 697, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\dGlsdisp \hspace{1cm} §11.5.7; 591, 697, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\dgls \hspace{1cm} §11.5.7; 183, 590, 595, 596, 696, 697, 698, 699, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\dglsdisp \hspace{1cm} §11.5.7; 591, 697, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\dGlspl \hspace{1cm} §11.5.7; 591, 698, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\dGlspl \hspace{1cm} §11.5.7; 591, 699, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\dglspl \hspace{1cm} §11.5.7; 590, 699, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\dglspl \hspace{1cm} §11.5.7; 593, 698
\DGlspl \hspace{1cm} §11.5.7; 591, 697, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\DGlspl \hspace{1cm} §11.5.7; 591, 698, see also \glsxtraddlabelprefix & \glsxtrprependlabelprefix
\digamma \hspace{1cm} 577–579
\display full form \hspace{1cm} 50, 63, 81, 88, 93, 100, 103, 104, 106, 109–111, 132, 170, 172, 173, 175, 625, 627, 846, 847
document environment \hspace{1cm} 16, 17, 37, 537, 604, 985, 986
\doifglossarynoexistsordo \hspace{1cm} 380, 699, see also \ifglossaryexists &
Index

\glsxtrifemptyglossary 307, 840
\dolistscloop 305, 307, 699, 841
\DTLformatlist 306, 532, 699, 758, 859, 862

E
\eglssetwidest §8.6.5.4; 439, 440, 700, 981
\eglsupdatewidest §8.6.5.4; 440, 700, 981
\emph 156, 158, 655–657, 660, 661, 663, 664
\entryname 451, 490, 499, 700
\Epsilon §11.5.8; 599, 700
\equationcounter 26, 27, 581, 582, 584, 840, 986
\equationenvironment 26, 27
\Eta §11.5.8; 599, 700
\etoolboxpackage 38–40, 307, 308, 321, 413, 530, 538, 640, 641, 695, 756, 840–842, 859, 869, 935
\exampleglossariesacronymbib §14.1; 607
\exampleglossariesacronymdescbib §14.1; 607
\exampleglossariesacronym-langbib §14.1; 607
\exampleglossariesbriefbib §14.1; 607
\exampleglossaries-childmultiparbib §14.1; 607
\exampleglossaries-childnonamebib §14.1; 607
\exampleglossaries-citebib §14.1; 607
\exampleglossaries-imagesbib §14.1; 607
\exampleglossaries-longbib §14.1; 607
\exampleglossaries-longchildbib §14.1; 608
\exampleglossaries-multiparbib §14.1; 608
\exampleglossaries-parentbib §14.1; 608
\exampleglossaries-symbolnamesbib §14.1; 608
\exampleglossaries-symbolsbib §14.1; 608
\exampleglossaries-urlbib §14.1; 608
\exampleglossaries-userbib §14.1; 608
\exampleglossaries-xrbib §14.1; 608
\exampleglossaries-xrtext §14.1; 607
\expandafter 166, 255
\expandonce 133, 134
\ExtraCustomAbbreviationFields §4.5.3.1; 162, 164, 700

F
\fancyhdrpackage 450
\fbox 243, 244
\figureenvironment 27
\fileformats
acr 6
aux 17, 26, 374, 412, 533, 538, 627, 640, 969, 985
bib 17, 33, 34, 533, 607, 628
glg-abr 9
glo-abr 9
gls-abr 9
gls §17
\glstex 34, 533, 536, 629
\glsdefs 16, 17, 37
\glslabels 993
\glstex 34, 533, 536, 629
\ldf 618, 619
\log 985
\text 33, 533, 607
\toc 215
firstuse 626
firstuseflag 626, 764

1001
Index

first use text ........................................... 626
\firstacronymfont .................................. 701
\FIRSTLC ........................................... §11.5.2; 561, 701
\FIRSTUC ........................................... §11.5.2; 561, 701
fontenc package ....................................... 61
\footnote ................................................. 64, 255
\forallabbreviationlists §8; 380, 701, see also \forallglossaries
\forallacronyms .......................................... 380, 701
\forallglossaries ........................................ 380, 701
\forallglsentries ........................................ 701
foreignlanguage ......................................... 296
\forgtlsentries ........................................ 15, 702, see also \forallglsentries
\forlistcsloop .......................................... 307, 841
\frontmatter ........................................... 188
full stop (.) 2, 12, 13, 248, 249, 435, 520, 522, 570, 836, 858, 862, 896, 991

G

gitlexstring package .................................... 215, 763
\GetTitleStringDisableCommands .................. 215
\GetTitleStringSetup ................................ 215
\gglssetwidest .......................................... §8.6.5.4; 439, 702
\gglsupdatewidest ..................................... §8.6.5.4; 440, 702
\gGlsXtrSetField ....................................... §3.5; 39, 40, 702
GUI ....................................................... 626

Glo

\glolinkprefix ......................................... 196, 257, 383, 415, 527, 652, 702
all ......................................................... 984
(name) .................................................. 984
glossaries-extra package §2: 2, 8, 984
abbreviations §2.1; 9, 10, 44, 46, 559, 618, 686, 820, 968, 969, 984
accsupp §2.3; 14, 168, 169, 421, 500, 631–637, 708, 709, 718–724, 729, 820, 821, 823, 898, 984
autoseeindex §2.4; 21, 22, 33, 267, 274, 279, 305, 984
false ..................................................... 22–24, 33, 635, 984
ture ...................................................... 22, 984
bibglsaux §2.4; 26, 691, 904, 985
debug §2.5; 28, 985, 990
all ......................................................... 29, 985
false ...................................................... 28, 985
showaccsupp ........................................... 28, 985
showtargets 29, 30, 796, 985, 988, see also showtargets
showwrgloss ........................................... 29, 934, 985
ture ...................................................... 28, 985
docdef §2.4; 16, 327, 373, 374, 604, 985
atom ..................................................... 17, 985
false ...................................................... 16, 985
restricted .............................................. 17, 985
ture ...................................................... 16, 37, 311, 379, 985
equations §2.4; 26, 581, 986
floats §2.4; 27, 986
indexcounter §2.4; 25, 27, 29, 584, 585,
Index

599, 640, 985, 986
indexcrossrefs §2.4; 20, 22, 23, 269, 270, 282, 986
false 20, 21, 24, 986
true 20, 21, 986
nomissingglstext §2.1; 9, 986
postdot §2.2; 2, 12, 427, 428, 889, 986,
see also nopostdot & postpunc
postpunc §2.2; 12, 13, 427, 428, 433, 788, 986
comma 13, 986
dot 13, 986
none 13, 986
⟨punctuation⟩ 986
prefix §2.3; 14, 54, 370, 986
record §2.4; 23, 987
alsoindex see hybrid
hybrid 23, 25, 29, 37, 535, 537, 558, 987
nameref 23, 24, 34, 196, 282, 375, 393, 535, 537, 558, 581, 627, 633, 819, 837, 987
off 23, 37, 389, 987
only 23, 24, 34, 37, 282, 375, 393, 537, 558, 633, 819, 987
shortcuts §2.4; 17–19, 311, 329, 987, 988
abbr see abbreviations
abbreviations Table 4.1; 18, 19, 56, 681, 682, 688–691, 835, 958, 987
abother 19, 987
ac Table 4.1; 18, 19, 43, 56, 682–685, 687, 688, 835, 958, 987
acother 19, 835, 987
acro see acronyms
acronyms 19, 682–685, 687, 688, 988
all 19, 20, 835, 988
false see all
none 20, 988
other 18, 835, 959, 960, 988
true see all
showtargets §2.5; 29, 384, 913, 988
anoteleft 31, 341, 988
annoteright 31, 988
innerleft 31, 988
innerright 31, 988
left 30, 988
right 30, 988
stylemods §2.2; 12–14, 388, 389, 391, 393, 432, 433, 443, 984, 988
all 13, 989
default 13, 14, 989
⟨list⟩ 14, 989
undefaction §2.4; 15, 16, 284, 286, 732, 862, 923, 937, 989
error 15, 24, 379, 380, 732, 858, 860, 989
warn 15, 17, 24, 39, 280, 379, 380, 536, 732, 742, 858, 860, 906, 989
glossaries–prefix package 14, 54, 55, 207–210, 218, 370, 616, 634, 635, 789, 957, 962–967, 987
glossaries package 2, 41, 989
acronym §2.1; 10, 11, 559, 686, 984, 989
acronymlists see acronyms
acronyms §2.1; 11, 44, 46, 559, 686, 968, 970, 984, 989, 990
automake 622, 888, 989
counter 27, 265, 554, 581, 631, 986, 990
debug 990
eentrycounter 380, 420, 422, 451, 462, 735, 790, 990
hyperfirst 187, 257, 523, 990
index §2.1; 11, 12, 428, 560, 893, 960, 969, 972, 990
indexonlyfirst 268, 524, 602, 937, 990
makeindex 582, 990
mfistuc 990
nogroupskip 434–436, 476, 756, 787, 937, 990
nolist 14, 990
nolong 990
nomain 6, 729, 990
nonumberlist 264, 402, 429, 475, 554, 628, 703, 785, 990
nopostdot §2.2; 2–4, 12, 13, 434, 788, 986, 991, see also postdot & postpunc
Index

noredefwarn 3, 991
nostyles 14, 433, 439, 443, 991
nosuper 991
notranslate 991
notree 439, 991
nowarn 703, 991
numberedsection 382, 489, 651, 704, 726, 991
numbers §2; 11, 19, 32, 428, 559, 786, 887, 893, 960, 969, 972, 991
sanitize 991
savenumberlist 539, 991
section 448, 704, 992
seenoidex 23, 992
error 992
ignore 992
warn 992
sort 393, 419, 992
clear 992
def 992
none 24, 992
standard 789, 991, 992
use 992
style 483, 992
subentrycounter 380, 420, 422, 452, 483, 790, 797, 992
symbols §2; 10, 18, 32, 428, 559, 587, 798, 887, 893, 960, 969, 973, 993
toc 2–4, 993
translate 3, 4, 991, 993
babel 993
false 993
true 993
writeglslabelnames 17, 993
writeglslabels 17, 993
xindy 264, 266, 431, 600, 993
\GlossariesAbbrStyleTooComplex%Warning 702
\GlossariesExtraInfo 593, 973
\glossariesextrasetup §2; 8, 14, 18, 559, 703, 977
\GlossariesExtraWarning 160, 703, 882, 888, 915, 924, 933, 957
\GlossariesExtraWarningNoLine 703
glossary–bookindex package §8.7.1; 14, 391, 443–450, 672, 828–831
glossary–inline package 434, 672, 760, 788, 789
glossary–list package 671–673, 763, 990
glossary–long package 677, 731, 786, 990
glossary–longbooktabs package 450, 482, 488, 673, 786, 787
glossary–longragged package 677
glossary–mcols package 677, 678
glossary–super package 678, 731, 786, 991
glossary–superragged package 678
glossary–table package §8.7.4; 482–499, 678, 798–804, 973, 983
glossary–topic package §8.7.3; 475–482, 678, 806–809
glossary–tree package 437, 439, 440, 449, 475, 477, 671, 672, 678, 679, 744, 795, 809–812, 991
glossary 626
glossary entry fields 640, 765, 766
childcount 483, 487, 492–494, 580, 640, 860
childlist 640, 799
indexcounter 599, 640, 864
loclist 35, 402, 538, 582, 640, 970
recordcount 547, 548, 640, 923
recordcount.(counter) 547, 548, 641, 899
recordcount.(counter).(location) 547, 548, 641, 871
record.(counter) 413, 414, 641, 819
secondarygroup 394, 395, 399, 401, 641
useri 467, 641, 765
userii 467, 641, 766
useriii 467, 641, 766
glossary entry keys 33, 631
access 162, 500, 501, 631, 718, 785
alias §3.2; 20, 33, 36, 37, 267, 269–271, 274, 275, 277, 279, 280, 282, 294, 328, 373, 607, 614, 631, 822, 903, 928
Index

<table>
<thead>
<tr>
<th>long–custom2-name</th>
<th>§8.7.2.7; 470–472, 673, 767</th>
</tr>
</thead>
<tbody>
<tr>
<td>long–custom3-name</td>
<td>§8.7.2.7; 471, 472, 673, 767</td>
</tr>
<tr>
<td>long–desc–custom1-name</td>
<td>§8.7.2.7; 473, 474, 673, 769</td>
</tr>
<tr>
<td>long–desc–custom2-name</td>
<td>§8.7.2.7; 474, 475, 673, 768</td>
</tr>
<tr>
<td>long–desc–custom3-name</td>
<td>§8.7.2.7; 474, 475, 674, 768</td>
</tr>
<tr>
<td>long–desc–name</td>
<td>§8.7.2.1; 455, 674, 769</td>
</tr>
<tr>
<td>long–desc–sym–name</td>
<td>§8.7.2.2; 457, 674, 770</td>
</tr>
<tr>
<td>long–loc–desc–name</td>
<td>§8.7.2.3; 459, 674, 771</td>
</tr>
<tr>
<td>long–loc–desc–sym–name</td>
<td>§8.7.2.4; 461, 674, 771</td>
</tr>
<tr>
<td>long–loc–sym–desc–name</td>
<td>§8.7.2.4; 460, 675, 771, 772</td>
</tr>
<tr>
<td>long–name–custom1</td>
<td>§8.7.2.7; 470, 471, 474, 675, 773, 774</td>
</tr>
<tr>
<td>long–name–custom1–desc</td>
<td>§8.7.2.7; 472–475, 675, 768, 773</td>
</tr>
<tr>
<td>long–name–custom2</td>
<td>§8.7.2.7; 470–472, 475, 675, 773, 774</td>
</tr>
<tr>
<td>long–name–custom2–desc</td>
<td>§8.7.2.7; 472–475, 675, 767, 773</td>
</tr>
<tr>
<td>long–name–custom3</td>
<td>§8.7.2.7; 471, 472, 475, 675, 774</td>
</tr>
<tr>
<td>long–name–custom3–desc</td>
<td>§8.7.2.7; 473–475, 675, 767, 774</td>
</tr>
<tr>
<td>long–name–desc–loc</td>
<td>§8.7.2.3; 458, 676, 774, 775</td>
</tr>
<tr>
<td>long–name–desc–sym–loc</td>
<td>§8.7.2.4; 459, 676, 775</td>
</tr>
<tr>
<td>long–name–desc–sym</td>
<td>§8.7.2.2; 456, 676, 775</td>
</tr>
<tr>
<td>long–name–desc</td>
<td>§8.7.2.1; 454, 455, 464, 474, 676, 677, 776</td>
</tr>
<tr>
<td>long–name–sym–desc–loc</td>
<td>§8.7.2.4; 460, 676, 776</td>
</tr>
<tr>
<td>long–name–sym–desc</td>
<td>§8.7.2.2; 454, 456, 1006</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>glossary, ignored</th>
<th>see ignored glossary</th>
</tr>
</thead>
<tbody>
<tr>
<td>glossary, mini</td>
<td>see mini-glossary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>abbr–long–short</th>
<th>§8.7.2.6; 466, 671, 772, 777–779</th>
</tr>
</thead>
<tbody>
<tr>
<td>abbr–short–long</td>
<td>§8.7.2.6; 466, 671, 772, 773, 777–779</td>
</tr>
<tr>
<td>altlist</td>
<td>434, 435, 671, 726</td>
</tr>
<tr>
<td>alttree</td>
<td>437–439, 441–443, 452, 475, 477, 580, 615, 671, 672, 726, 823</td>
</tr>
<tr>
<td>alttreegroup</td>
<td>671</td>
</tr>
<tr>
<td>bookindex</td>
<td>§8.7.1; 14, 393, 398, 443–450, 559, 672, 828–831</td>
</tr>
<tr>
<td>index</td>
<td>437, 672, 677, 809–813</td>
</tr>
<tr>
<td>indexgroup</td>
<td>672, 760, 810, 812</td>
</tr>
<tr>
<td>inline</td>
<td>434, 672</td>
</tr>
<tr>
<td>list</td>
<td>14, 434, 435, 671, 672, 673, 762–764, 983</td>
</tr>
<tr>
<td>listdotted</td>
<td>434, 435, 672, 763</td>
</tr>
<tr>
<td>listgroup</td>
<td>435, 672, 673, 763</td>
</tr>
<tr>
<td>listhypergroup</td>
<td>673</td>
</tr>
<tr>
<td>long–booktabs</td>
<td>483, 673</td>
</tr>
<tr>
<td>long–custom1–name</td>
<td>§8.7.2.7; 470, 471, 673, 767, 768</td>
</tr>
</tbody>
</table>

textaccess .................................. 162, 502, 637, 721, 805

type ........................................ 41, 44, 255, 265, 379, 380, 391, 420, 431, 517, 562, 637, 696, 739, 887, 940, 959, 961

user1 63, 64, 135, 298, 331, 369, 370, 399, 425, 507, 514, 520, 628, 637, 641, 722, 739, 815

user1access .................................. 637, 722, 815

user2 296, 369, 370, 507, 515, 638, 641, 722, 740, 815

user2access .................................. 638, 722, 815

user3 508, 515, 638, 641, 722, 740, 816

user3access .................................. 638, 722, 816

user4 508, 516, 638, 723, 740, 817

user4access .................................. 638, 723, 817

user5 509, 516, 638, 639, 723, 741, 817

user5access .................................. 639, 723, 817

user6 509, 516, 639, 724, 741, 818

user6access .................................. 639, 724, 818
Index

462, 676, 776, 777
long-sym-desc-name §8.7.2.2; 457, 677, 780, 781
long-sym-desc §8.7.2.5; 454, 463, 677, 780, 781
long ................. 14, 404, 434, 451, 677
longheader ........... 483, 677
longragged .......... 677
mcolindex ............ 677
mcolindexgroup ....... 443, 677
mcoltree ............. 476, 678
super ................ 678
superragged ........... 678
table ................. §8.7.4; 482, 483, 678, 973
topic ................. §8.7.3; 475, 476, 678
topicmcols ........... §8.7.3; 476, 478, 806
tree .................. 437, 438, 475, 476, 671, 678, 679, 809–813, 923
treegroup ............. 435, 678, 679, 810, 812
treehypergroup ....... 436, 679, 811
treenoname .......... 438, 679, 811, 812
glossaryentrycounter .. 978
\glossaryentrynumbers 402, 429, 703, 845
\glossaryheader ....... 403, 703
\glossaryname ........ 704
\glossarypostamble ... 386, 483, 651, 704
\glossarypreamble .... 380, 386, 387, 483, 651, 704
\glossarysection ..... 483, 704
\glossarytitle ......... 403, 489, 704
\glossarytocname .... 403, 704
\glossentry ...... 402, 403, 405, 419, 483, 704
\Glossentrydesc ..... 159, 481, 705, 730
\glossentrydesc .... 159, 424, 452, 497, 524, 525, 631, 705, 730
\Glossentryname ..... 425, 480, 600, 705, 784
\glossentryname .... 159, 419–421, 424, 425, 445, 452, 497, 502, 526, 600, 629, 634, 705, 785, 804, 851, 897
\glossentrynameother §8.6; 159, 421, 425, 705, 897
\Glossentrysymbol .... 159, 705, 797
\glossentrysymbol 159, 424, 425, 453, 497, 526, 705, 736, 798
\glossxtrsetpopts ... §5.4; 231, 705, 891
\GLS .......................... 706
\Gls .......................... 706
\gs .......................... 706
\gls-like ................ 626
\gls-like and \glslike-options 186–196, 642
counter ............. 27, 265, 548, 554, 583, 584, 642, 959
format .................. 642
hyper ................. 183, 186, 187, 189, 204, 205, 231, 257, 323, 324, 523, 524, 642, 832
hyperoutside ....... §5.1.2; 186, 190, 235, 257, 524, 642, 864
innertextformat ... §5.1.2; 43, 166, 174, 191, 627, 642, 834, 835
local .................. 191, 286, 643
noindex ............... §5.1.2; 37, 51, 185, 186, 188, 190, 193, 194, 204, 205, 231, 257, 260, 268, 281, 282, 602, 643, 860
postunset ............. §5.1.2; 186, 189–193, 286, 643, 760
prefix ................. §5.1.2; 196, 385, 643
prereset ............ §5.1.2; 43, 47, 185, 186, 189, 191, 193, 285, 286, 290, 364, 643, 745, 748, 749, 760
prefix .................. §5.1.2; 196, 385, 643
preunset .......... §5.1.2; 43, 47, 185, 186, 189, 192, 193, 285, 286, 288, 289, 365, 643, 760, 787, 788, 805
textformat ........... §5.1.2; 191, 240, 643
theHvalue ........... §5.1.2; 195, 196, 644
thevalue §5.1.2; 195, 196, 265, 528, 580, 616, 625, 644
types .................. 186, 644, 725
wrgloss §5.1.2; 185, 186, 189, 190, 194, 267, 524, 644, 865, 938
\glsabbrvdefaultfont 66, 74, 82, 93, 115, 132, 133, 136, 148, 154, 706
\glsabbrvemfont ...... 71–73, 78–80, 85–87, 97–99, 126–129, 158, 706

1007
Index

\GLSaccesslong \ §9.2; 506, 711, 717
\Glsaccesslong \ §9.2; 506, 711, 717
\glaccesslong \ §9.2; 176, 506, 711, 717, 718
\GLSaccesslongpl \ §9.2; 507, 712, 718
\Glsaccesslongpl \ §9.2; 507, 712, 718
\glaccesslongpl \ §9.2; 507, 712, 718
\GLSaccessname \ §9.2; 502, 712, 718
\Glsaccessname \ §9.2; 502, 712, 718
\glaccessname \ §9.2; 502, 712, 718
\GLSaccessplural \ §9.2; 503, 712, 719
\Glsaccessplural \ §9.2; 503, 713, 719
\glaccessplural \ §9.2; 503, 713, 719
\GLSaccessshort \ §9.2; 506, 713, 719
\Glsaccessshort \ §9.2; 506, 713, 719
\glaccessshort \ §9.2; 177, 505, 713, 719
\GLSaccessshortpl \ §9.2; 506, 713, 719
\Glsaccessshortpl \ §9.2; 506, 713, 719
\glaccessshortpl \ §9.2; 506, 713, 720
\GLSaccesssymbol \ §9.2; 504, 714, 720
\Glsaccesssymbol \ §9.2; 504, 714, 720
\glaccesssymbol \ §9.2; 504, 714, 720
\GLSaccesssymbolplural \ §9.2; 505, 714, 720
\Glsaccesssymbolplural \ §9.2; 504, 714, 720
\glaccesssymbolplural \ §9.2; 504, 714, 720
\GLSaccessuserv \ §9.2; 509, 717, 723
\Glsaccessuserv \ §9.2; 509, 717, 723
\glaccessuserv \ §9.2; 509, 717, 723
\GLSaccessuservi \ §9.2; 509, 717, 723
\Glsaccessuservi \ §9.2; 509, 717, 723
\glaccessuservi \ §9.2; 509, 717, 723
\GLSaccesssymbolplural \ §9.2; 506, 713, 719, 720
\Glsaccesssymbolplural \ §9.2; 506, 713, 719, 720
\glaccesssymbolplural \ §9.2; 506, 713, 719, 720
\GLSaccesslongpl \ §9.2; 507, 712, 718
\Glsaccesslongpl \ §9.2; 507, 712, 718
\glaccesslongpl \ §9.2; 507, 712, 718
\GLSaccessname \ §9.2; 502, 712, 718
\Glsaccessname \ §9.2; 502, 712, 718
\glaccessname \ §9.2; 502, 712, 718
\GLSaccessplural \ §9.2; 503, 712, 719
\Glsaccessplural \ §9.2; 503, 713, 719
\glaccessplural \ §9.2; 503, 713, 719
\GLSaccessshort \ §9.2; 506, 713, 719
\Glsaccessshort \ §9.2; 506, 713, 719
\glaccessshort \ §9.2; 177, 505, 713, 719
\GLSaccessshortpl \ §9.2; 506, 713, 719
\Glsaccessshortpl \ §9.2; 506, 713, 719
\glaccessshortpl \ §9.2; 506, 713, 720
\GLSaccesssymbol \ §9.2; 504, 714, 720
\Glsaccesssymbol \ §9.2; 504, 714, 720
\glaccesssymbol \ §9.2; 504, 714, 720
\GLSaccesssymbolplural \ §9.2; 505, 714, 720
\Glsaccesssymbolplural \ §9.2; 504, 714, 720
\glaccesssymbolplural \ §9.2; 504, 714, 720
\GLSaccessuserv \ §9.2; 509, 717, 723
\Glsaccessuserv \ §9.2; 509, 717, 723
\glaccessuserv \ §9.2; 509, 717, 723
\GLSaccessuservi \ §9.2; 509, 717, 723
\Glsaccessuservi \ §9.2; 509, 717, 723
\glaccessuservi \ §9.2; 509, 717, 723

\glsxtrprovidestoragekey
\glxtrprovidestoragekey
\glsaddkey
\glsaddkey &
\glsaddstoragekey
\glsaddstoragekey &
\glsaddstoragekey

\glsaddall
\glsaddall
\glsaddall
\glsaddall
\glsaddall
\glsaddall
\glsaddall
\glsaddall
\glsaddall
\glsaddall
\glsaddall

\glsaddkey
\glsaddkey
\glsaddkey
\glsaddkey

\glsaddpostsetkeys
\glsaddpostsetkeys
\glsaddpostsetkeys
\glsaddpostsetkeys

\glsaddstoragekey
\glsaddstoragekey
\glsaddstoragekey
\glsaddstoragekey

\glsaltlistitem
\glsaltlistitem
\glsaltlistitem
\glsaltlistitem

\glsalttreechildpredesc
\glsalttreechildpredesc
\glsalttreechildpredesc
\glsalttreechildpredesc

\glscapitalisewords
\glscapitalisewords
\glscapitalisewords
\glscapitalisewords

\gscapecase
\gscapecase
\gscapecase
\gscapecase

\gsccapturedgroup
\gsccapturedgroup
\gsccapturedgroup
\gsccapturedgroup

\gscategory
\gscategory
\gscategory
\gscategory

\gscategorylabel
\gscategorylabel
\gscategorylabel
\gscategorylabel

\glscombinedfirstsep
\glscombinedfirstsep
\glscombinedfirstsep
\glscombinedfirstsep

\glscombinedsep
\glscombinedsep
\glscombinedsep
\glscombinedsep

\glscurrententrylabel
\glscurrententrylabel
\glscurrententrylabel
\glscurrententrylabel

1009
Index

405–407, 419, 426, 428, 728
\glscurrententrylevel \ §8.4.3; 396, 405, 407, 728
\glscurrentfieldvalue ... 304–309, 728, 841, 845, 858–860, 863, 936
\glscurrentrootentry \ §8.4.3; 405, 407, 729
\glscurrenttoplevelentry \ §8.4.3; 405, 407, 729
\glscustomtext ... 255, 729, 761
\glsdefaulttext ... 501, 729
\glsdefaulttype ... 44, 381, 653, 690, 696, 702, 729, 940, 968, 977
\glsdefpostdesc \ §8.6.2; 428, 729
\glsdefpostlink \ §5.5.4; 251, 729
\glsdefpostname \ §8.6.1; 426, 730
GLSdesc ... 730
Glsdesc ... 730
\glsdesc ... 253, 631, 730
\glsdescplural ... 253, 730
\glsdescriptionaccessdisplay ... 505, 512, 730
\glsdescriptionpluralaccessdisplay ... 505, 513, 731
\glsdescwidth ... 452, 454, 456, 458, 459, 462, 464, 466, 472, 473, 731, 767, 768, 772, 777, 778, 781
\Glsdisp ... 591, 731
\glsdisp ... 183, 236, 243, 255, 259, 369, 591, 626, 697, 729, 731, 762, 839, 886, 955
\glsdisplaynumberlist \ §11.5.8; 442, 598, 736
\glsdoifexists ... 731, 732, 742, 935, see also
  \ifglsentryexists,
  \glsdoifexistssordo &
  \glsdoifnoexists
\glsdoifexistssordo ... 732, see also
  \ifglsentryexists,
  \glsdoifexistssordo &
  \glsdoifnoexists
\glsdoifexistsorwarn ... 732
\glsdoifnoexists ... 732, see also
  \ifglsentryexists,
  \glsdoifexistssordo &
  \glsdoifnoexists
\glsenableentrycount ... 283, 310–312, 316, 526, 732, 733, 737, 896, 897
\glsenableentryunitcount \ §6.1; 283, 316, 732, 733, 737, 896, 897
\glsencapwrcontent \ §5.8; 185, 267, 732
\glsendrange \ §5.8; 262, 263, 733, 905
\glsentrycounterlabel ... 733
\glsentrycurrcount ... 310, 317, 733
Glsentrydesc ... 525, 733
\glsentrydesc ... 419, 505, 630, 631, 708, 733
\glsentrydescplural ... 505, 709, 733
Glsentryfirst ... 734
\glsentryfirst ... 213, 503, 709, 734
\glsentryfirstplural ... 734
\glsentryfirstplural ... 214, 504, 709, 734
\glsentryfmt ... 159, 255, 519, 696, 734, 755, 820, 848, 900
\glsentryindexcount \ §5.8; 268, 734, 758
\glsentryitem ... 420, 733, 735
Glsentrylong ... 50, 735
\glsentrylong ... 50, 208, 209, 229, 234, 506, 718, 735, 786
\glsentrylongpl ... 735
\glsentrylongpl ... 50, 209, 210, 507, 718, 735, 786
\glsentryname ... 377, 423, 735
\glsentryname ... 23, 47, 212, 294, 295, 303, 377, 419, 423, 502, 517, 600, 601, 630, 634, 718, 736
\glsentrynumberlist \ §11.5.8; 442, 598, 736
\glsentryparent ... 295, 736
\glsentrypdfsymbol \ §8.6; 424, 736
Glsentryplural ... 736
\glsentryplural ... 213, 503, 719, 736
\glsentryprevcount ... 310, 317, 737
\glsentryprevmaxcount \ §6.1; 317, 737
\glsentryprevtotalcount \ §6.1; 317, 737
Glsentryshort ... 50, 737
\glsentryshort ... 50, 202, 207, 216, 229, 230, 234, 506, 719, 737, 786
\glsentryshortpl ... 737
\glsentryshortpl ... 50, 207, 208, 210, 506, 720, 738, 786
Index

\Glsentrysymbol ........................................ 738
\glsentrysymbol ........................................ 424, 504, 636, 720, 738
\glsentrysymbolaccess ................................ 738
\glsentrysymbolplural 504, 637, 721, 738
\Glsentrytext ........................................ 725, 738
\glsentrytext ........................................ 15, 50, 202, 212, 230, 303, 503, 721, 725, 739, 739
\glsentrytitlecase .................................. 200, 377, 525, 739
\Glsentrytype ........................................ 739
\Glsentryuseri ........................................ 739
\glsentryuseri ........................................ 507, 637, 722, 739
\Glsentryuserii ....................................... 739
\glsentryuserii .................................... 507, 638, 722, 739
\Glsentryuseriii .................................... 740
\glsentryuseriii .................................... 508, 638, 723, 740
\Glsentryuseriv .................................... 740
\glsentryuseriv .................................... 508, 638, 723, 740
\Glsentryuserv ....................................... 741
\glsentryuserv ....................................... 509, 638, 723, 741
\Glsentryuservi ..................................... 741
\glsentryuservi ..................................... 509, 639, 724, 741
\glsexclapplyinnerfmtfield ................. §5.5.3; 162, 246, 741
\glextrapolpostnamehook §8.6.1; 426, 629, 742, 897
\glsfieldaccsupp ..................................... 742, 832
\glsfielddef ........................................ 37, 742, 743
\glsfielddef ........................................ 742, 743
\glsfieldfetch ....................................... 739, 742
\glsfieldgdef ........................................ 743
\glsfieldgdef ........................................ 743
\glsfieldxdef ........................................ 743
\glsFieldaccessdisplay §8.6.5.4; 441, 743
\glsFindWidestAnyName §8.6.5.4; 441, 743
\glsFindWidestAnyNameLocation §8.6.5.4; 442, 743
\glsFindWidestAnyNameSymbol §8.6.5.4; 441, 744
\glsFindWidestAnyNameSymbolLocation §8.6.5.4; 442, 743
\glsFindWidestLevelTwo §8.6.5.4; 441, 743
\glsFindWidestTopLevelName §8.6.5.4; 440, 744
\glsfindwidesttoplevelname 440, 744
\glsexclapplyinnerfmtfield §5.5.3; 162, 246, 741
\glextrapolpostnamehook §8.6.1; 426, 629, 742, 897
\glsexclapplyinnerfmtfield §5.5.3; 162, 246, 741
\glsextrapolpostnamehook §8.6.1; 426, 629, 742, 897
\glsfieldaccsupp ..................................... 742, 832
\glsfielddef ........................................ 37, 742, 743
\glsfielddef ........................................ 742, 743
\glsfieldfetch ....................................... 739, 742
\glsfieldgdef ........................................ 743
\glsfieldgdef ........................................ 743
\glsfieldxdef ........................................ 743
\glsFieldaccessdisplay §8.6.5.4; 441, 743
\glsFindWidestAnyName §8.6.5.4; 441, 743
\glsFindWidestAnyNameLocation §8.6.5.4; 442, 743
\glsFindWidestAnyNameSymbol §8.6.5.4; 441, 744
\glsFindWidestAnyNameSymbolLocation §8.6.5.4; 442, 743
\glsFindWidestLevelTwo §8.6.5.4; 441, 743
\glsFindWidestTopLevelName §8.6.5.4; 440, 744
\glsfindwidesttoplevelname 440, 744
Index

\glsifattribute \textbullet\ §10.2.2; 530, 757
\glsifattributetrue \textbullet\ §10.2.2; 531, 758
\glsifcategory \textbullet\ §10; 517, 758
\glsifcategoryattribute \textbullet\ §10.2.2; 530, 531, 758
\glsifcategoryattributehasitem \textbullet\ §10.2.2; 532, 758
\glsifcategoryattributetrue \textbullet\ §10.2.2; 531, 758
\glsifindexed \textbullet\ §5.8; 268, 758
\glsifnotregular \textbullet\ §10.2.2; 531, 759
\glsifnotregularcategory \textbullet\ §10.2.2; 531, 759
\glsifplural \textbullet\ 142, 144, 150, 152, 255, 354, 355, 759
\glsifregular \textbullet\ §10.2.2; 256, 531, 734, 759
\glsifregularcategory \textbullet\ §10.2.2; 531, 759
\glsindexingsetting \textbullet\ §5.8; 268, 758
\glsifnotregular \textbullet\ §10.2.2; 531, 759
\glsignore \textbullet\ 265, 429, 550, 627, 759
\glsindexingsetting \textbullet\ §5.8; 268, 758
\glsindexsubgroupitem \textbullet\ §8.6.5.3; 435, 760
\glsinitreunsets \textbullet\ §5.1.1; 186, 189, 289, 760
\glsinlinedescformat \textbullet\ 434, 760
\glsinlinesubdescformat \textbullet\ 434, 760
\glsinnerfmtabbrvfont \textbullet\ §4.5.3.1; 167, 168, 760, 818
\glsinnerfmtlongfont \textbullet\ §4.5.3.1; 167, 168, 760, 818
\glsinsert \textbullet\ 54, 105, 150–152, 166, 254, 255, 752, 756, 847, 901, see also \glsxtrsaveinsert & \GlsXtrIfUnusedOrUndefined
\glskeylisttok \textbullet\ §4.5.3.1; 161–163, 761
\glslabel \textbullet\ 150–152, 165, 166, 230, 244, 247, 255, 263, 289, 527, 761, 824, 895
\glslabeltok \textbullet\ §4.5.3.1; 161, 163–166, 761
\glsletentryfield \textbullet\ 294, 761
\Glslink \textbullet\ 183, 236, 243, 253, 259, 299, 591, 626, 698, 731, 761, 843, 887
\glslinkcheckfirsthyperhook \textbullet\ 187, 310, 319, 322, 762
\glslinkpostsetkeys \textbullet\ 187–190, 310, 323, 762
\glslinkpresetkeys \textbullet\ §5.1.1; 187–190, 310, 322, 323, 558, 762
\glslinkwrcontent \textbullet\ §5; 185, 230, 732, 762
\glslistchildpostlocation \textbullet\ §8.6.5.3; 435, 762
\glslistchildprelocation \textbullet\ §8.6.5.3; 434, 762
\glslistdesc \textbullet\ §8.6.5.3; 434, 763
\glslistdottedwidth \textbullet\ §8.6.5.3; 434, 763
\glslistexpandedname \textbullet\ 763
\glslistgroupafterheader \textbullet\ §8.6.5.3; 435, 763
\glslistgroupheaderitem \textbullet\ §8.6.5.3; 435, 763
\glslistgroupskip \textbullet\ §8.6.5.3; 435, 763
\glslistinit \textbullet\ §5.10; 284, 764, see also \glsreset, \glslocalunset, \ifglsused & \GlsXtrIfUnusedOrUndefined
\glslocalreset \textbullet\ 192, 283, 764, 896, see also \glsreset, \glslocalunset, \ifglsused & \GlsXtrIfUnusedOrUndefined
\glslocalresetall \textbullet\ 290, 764, see also \glsreset, \glslocalreset, \glsresetall & \glslocalunsetall
\glslocalreseteach \textbullet\ §5.10; 284, 764, see also \glsreset, \glslocalreset, \ifglsused & \GlsXtrIfUnusedOrUndefined
\glslocalunset \textbullet\ 283, 286–288, 643, 764, 896, see also \glsunset, \glslocalreset, \ifglsused & \GlsXtrIfUnusedOrUndefined
\glslocalunsetall \textbullet\ 764, see also \glsunset, \glslocalunset, \ifglsused & \GlsXtrIfUnusedOrUndefined
\glslongaccessdisplay \textbullet\ 506, 514, 765
\glslongdefaultfont \textbullet\ 65, 74, 81, 93, 133,
Index

TabularFooter . §8.7.2.3; 459, 771
\gls{longextraLocationDescName}TabularHeader . §8.7.2.3; 459, 771
\gls{longextraLocationDescSymName}TabularHeader . §8.7.2.4; 461, 771
\gls{longextraLocationDescSymName}TabularFooter . §8.7.2.4; 461, 771
\gls{longextraLocationDescSymName}TabularHeader . §8.7.2.4; 461, 772
\gls{longextraLocationFmt}§8.7.2; 453, 771
\gls{longextraLongHeader}§8.7.2.6; 464, 772
\gls{longextraLongShortHeader}§8.7.2.6; 467, 772
\gls{longextraLongShortTabularFooter}§8.7.2.6; 466, 772
\gls{longextraLongShortTabularHeader}§8.7.2.6; 466, 772
\gls{longextraNameAlign}§8.7.2; 452, 464, 773
\gls{longextraNameCustomIDesc}TabularHeader . §8.7.2.7; 453, 772
\gls{longextraNameCustomIDesc}TabularHeader . §8.7.2.7; 453, 773
\gls{longextraNameCustomIDesc}TabularFooter . §8.7.2.7; 460, 775
\gls{longextraNameCustomITabular}TabularHeader . §8.7.2.7; 470, 774
\gls{longextraNameCustomIIITabular}TabularHeader . §8.7.2.7; 474, 774
\gls{longextraNameCustomIIITabular}TabularFooter . §8.7.2.7; 474, 774
Index

\glslongextraNameSymDescTabularFooter .......... §8.7.2.2; 457, 776
\glslongextraNameSymDescTabularHeader .......... §8.7.2.2; 456, 777
\glslongextraSetDescWidth .......... §8.7.2.1;
454, 456, 458, 472, 777
\glslongextraSetWidest §8.7.2; 452–454, 456, 458, 459, 777
\glslongextraShortHeader §8.7.2.6; 464, 466, 777
\glslongextraShortLongHeader §8.7.2.6; 466, 777
\glslongextraShortLongTabularFooter §8.7.2.6;
466, 777
\glslongextraShortLongTabularHeader §8.7.2.6; 466, 777
\glslongextraShortTargetFmt §8.7.2.6;
466, 777
\glslongextraSubCustomIFmt §8.7.2.7; 468, 471, 778
\glslongextraSubCustomIIIFmt §8.7.2.7; 469, 472, 778
\glslongextraSubDescFmt .......... §8.7.2;
452, 474, 778
\glslongextraSubGroupHeading §8.7.2;
453, 778
\glslongextraSubLocationFmt §8.7.2;
453, 473, 779
\glslongextraSubLongFmt .......... §8.7.2.6;
465, 779
\glslongextraSubNameFmt .......... §8.7.2;
452, 453, 462, 471, 779
\glslongextraSubShortTargetFmt §8.7.2.6;
465, 779
\glslongextraSubSymbolFmt .......... §8.7.2;
453, 779
\glslongextraSubSymbolOrName §8.7.2.5;
462, 779
\glslongextraSubSymbolTargetFmt §8.7.2.5;
462, 779
\glslongextraSubSymbolAlign .......... §8.7.2;
454, 780
\glslongextraSymbolFmt §8.7.2; 453, 454, 462, 780
\glslongextraSymbolNameAlign §8.7.2.5;
462, 780
\glslongextraSymbolTargetFmt §8.7.2.5;
462, 780
\glslongextraSymbolAlign .......... §8.7.2;
454, 780
\glslongextraSymbolFmt §8.7.2; 453, 454, 462, 780
\glslongextraSymbolNameAlign §8.7.2.5;
462, 780
\glslongextraSymbolTargetFmt §8.7.2.5;
462, 780
\glslongextraSymbolAlign .......... §8.7.2;
454, 780
\glslongextraSymDescNameHeader .......... §8.7.2.2;
457, 780
\glslongextraSymDescTabularFooter .......... §8.7.2.2;
457, 781
\glslongextraSymDescTabularHeader .......... §8.7.2.2;
457, 781
\glslongextraSymDescNameTabularFooter .......... §8.7.2.2;
457, 781
\glslongextraSymDescNameTabularHeader .......... §8.7.2.2;
457, 781
\glslongextraSymDescNameTabularFooter .......... §8.7.2.2;
457, 781
\glslongextraSymDescNameTabularFooter .......... §8.7.2.2;
457, 781
\glslongextraSymDescNameTabularFooter .......... §8.7.2.2;
457, 781
\glslongextraSymDescNameTabularFooter .......... §8.7.2.2;
457, 781
\glslongextraSymDescNameTable .......... §8.7.2.2;
457, 781
\glslongextraSymLocSetDescWidth §8.7.2.4;
459, 781
\glslongextraSymNoNameSetDescWidth §8.7.2.5;
462, 781
\glslongextraSymSetDescWidth §8.7.2.2;
456, 459, 781
\glslongextraSymDescHeader .......... §8.7.2.5;
463, 780
\glslongextraSymDescTabularFooter .......... §8.7.2.5;
463, 781
\glslongextraSymDescTabularFooter .......... §8.7.2.5;
463, 781
\glslongextraSymDescTabularFooter .......... §8.7.2.5;
463, 781
\glslongextraSymDescTabularFooter .......... §8.7.2.5;
463, 781
\glslongextraSymDescTabularFooter .......... §8.7.2.5;
463, 781
\glslongextraSymDescTabularFooter .......... §8.7.2.5;
463, 781
\glslongextraSymDescTabularFooter .......... §8.7.2.5;
463, 781
\glslongextraSymDescTabularFooter .......... §8.7.2.5;
463, 781
\glslongextraTabularVAlign .......... §8.7.2;
451, 782
\glslongextraUpdateWidest .......... §8.7.2;
453, 782
\glslongextraUpdateWidestChild §8.7.2;
453, 782
\GlsLongExtraUseTabularfalse .......... §8.7.2;
451, 782
\GlsLongExtraUseTabulartrue .......... §8.7.2;
450, 451, 782
\glslongfont .......... 168, 172, 760, 782, 819
\glslongfootnotefont .......... 115, 145, 782
\glslonghyphenfont 64, 103, 108, 148, 783
\glslongonlyfont .......... 111, 154, 783
\glslongpltok §4.5.3.1; 162, 164, 783
\glslongpluralaccessdisplay .......... 507,
Index

514, 783
\glslongtok . . . §4.5.3.1; 156, 161, 164, 783
\glslonguserfont . . . . 88, 100, 136, 783
\glslowercase . . . §5.2.2; 200, 783
\glsmakefirstuc . . . . 783
\glsmfuadmap . . . §5.2.1; 199, 200, 295, 593, 784
\glsmfublocker . . . §5.2.1; 199, 593, 784
\glsfuexcl . . . §5.2.1; 198–200, 784
\GLSNam e . . . 81, 89, 91, 93, 101, 103, 212, 784
\GLSname . . . 212, 368, 784, 951
\glcsname . . . 212, 355, 368, 634, 784, 951
\glcsnameaccessdisplay . . . 502, 510, 785
\glcsnamefont . . . 420, 425, 464, 526, 785
\glsnextpages . . . 421, 785
\glsnoexpandfields . . . 205, 579, 785, 794
\glsoidxdisplayloc . . . §11.5.6; 430, 582, 785, 837
\glsnonextpages . . . 421, 785, 791
\glsnumberformat . . . 188, 628, 785
\glsnumbersgroupname . . . 432, 786, 991
\glspagelistwidth . . . 453, 458, 459, 786
\glspar . . . 786
\glspatchLToutput . . . 786
\glspdfmtfull . . . §5.3.2; 210, 211, 751, 786
\glspdfmtfullpl . . . §5.3.2; 210, 211, 751, 752, 786
\glspenaltygroupsdisplay . . . 482, 787
\glspenaltychars . . . 527, 787
\GLSp l . . . 787
\GLSp l . . . 787
\glspl . . . 787
\GLSplural . . . 213, 787
\GLSplural . . . 213, 787
\glsplural . . . 43, 47, 81, 185, 192, 213, 241, 253, 520, 759, 787, 788
\glspluralaccessdisplay . . . 503, 510, 788
\glspluralsuffix . . . 35, 36, 43, 44, 162, 632, 634, 788
\glspostdescription . . . 12, 13, 426–428, 433, 434, 629, 631, 788, 893
\glspostinline . . . 434, 788
\glspostlinelines . . . 789
\glspostlinelines . . . 789
\glspostlinkhook . . . 231, 247, 350, 352, 629, 789
\glsprefixsep . . . 55, 789
\glsprestandardsort . . . 789, 992
\glspretopostlink . . . §5.5.4; 251, 789
\GLSp s . . . §5.4; 232, 790
\Glsps . . . §5.4; 232, 790
\glsps . . . §5.4; 232, 335, 790
\GLSp t . . . §5.4; 232, 790
\Glspt . . . §5.4; 232, 790
\glspt . . . §5.4; 232, 790
\glsrefentry . . . 380, 422, 790, 891
\glsreset . . . 192, 283, 310, 791, 897, see also \glslocalreset, \glsunset,
\ifglsused & \GlsXtrIfUnusedOrUndefined
\glsresetall . . . 283, 791, see also \glsreset,
\glsunsetall, \ifglsused & \GlsXtrIfUnusedOrUndefined
\glsresetcurrcountfalse . . . §6.1; 310, 311, 791
\glsresetcurrcounttrue . . . §6.1; 311, 791
\glsresetentrylist . . . 404, 791
\glssee . . . 21, 22, 24, 33, 37, 186, 260, 261, 269, 270, 274, 275, 281, 635, 791, see also \glsxtrindexseealso
\glsseefirstitem . . . §5.13; 303, 791, 792, see also \glsseefirstitem
\glsseeformat . . . 281, 302, 792, 933
\glsseeitem . . . 302–304, 792, see also \glsseeitem
\glsseeformat . . . §5.13; 295, 303, 792
\glsseelastoxfordsep . . . §5.13; 295, 303, see also \glsseelastoxfordsep
\glssee . . . §5.13; 304, 792, see also \glssee
\glsseelastoxfordsep . . . §5.13; 304, 690, 792, see also \glsseelastoxfordsep
\glsseelist . . . §5.13; 303, 305, 792, 793, 903, 917, see also \glsxtrseelist
\glssee . . . §5.13; 304, 792, 793, see also \glssee
\glsseelastsep . . . \glsseelastsep & \glsseelastoxfordsep
\glsxtrindexseealso
\glsxtrifundefined\glsxtrifundefined

1017
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>\glssentencecase</td>
<td>§5.2.1; 197, 198, 731, 761, 793</td>
</tr>
<tr>
<td>\glssubgroupheading</td>
<td>§8.4.1; 398, 405, 406, 797</td>
</tr>
<tr>
<td>\glssetabbrvfmt</td>
<td>§4.5.2; 159, 170, 256, 696, 734, 793</td>
</tr>
<tr>
<td>\glssetattribute</td>
<td>§10.2.2; 529, 793</td>
</tr>
<tr>
<td>\glssetcategoriesattribute</td>
<td>§10.2.2; 529, 793</td>
</tr>
<tr>
<td>\glssetcategoriesattributes</td>
<td>§10.2.2; 529, 794</td>
</tr>
<tr>
<td>\glssetcategoryattribute</td>
<td>§10.2.2; 529, 549, 793</td>
</tr>
<tr>
<td>\glssetcategoryattributes</td>
<td>§10.2.2; 529, 793</td>
</tr>
<tr>
<td>\glssetcombinedsepabbrvnbsp</td>
<td>§7.4; 348, 794</td>
</tr>
<tr>
<td>\glssetcombinedsepabbrvnone</td>
<td>§7.4; 348, 794</td>
</tr>
<tr>
<td>\glssetcombinedsepnarrow</td>
<td>§7.4; 349, 794</td>
</tr>
<tr>
<td>\glssetexpandfield</td>
<td>785, 794</td>
</tr>
<tr>
<td>\glssetnoexpandfield</td>
<td>742, 794</td>
</tr>
<tr>
<td>\glssetregularcategory</td>
<td>§10.2.2; 529, 795</td>
</tr>
<tr>
<td>\glssetwidest</td>
<td>439, 452, 475, 580, 671, 700, 702, 795, 814</td>
</tr>
<tr>
<td>\glsshortaccessdisplay</td>
<td>506, 513, 795</td>
</tr>
<tr>
<td>\glsshortacssupp</td>
<td>742, 795, 898</td>
</tr>
<tr>
<td>\glsshortpltok</td>
<td>§4.5.3.1; 162, 164, 167, 795</td>
</tr>
<tr>
<td>\glsshortpluralaccessdisplay</td>
<td>506, 513, 795</td>
</tr>
<tr>
<td>\glsshorttток</td>
<td>§4.5.3.1; 145, 161, 162, 164, 167, 795</td>
</tr>
<tr>
<td>\glswift</td>
<td>796</td>
</tr>
<tr>
<td>\glswiftfont</td>
<td>796</td>
</tr>
<tr>
<td>\glswiftfonttext</td>
<td>28, 796</td>
</tr>
<tr>
<td>\glswiftinner</td>
<td>29, 30, 796, 913</td>
</tr>
<tr>
<td>\glswiftinnersonsymleft</td>
<td>§2.5; 30, 796</td>
</tr>
<tr>
<td>\glswiftinnersonsymright</td>
<td>§2.5; 30, 796</td>
</tr>
<tr>
<td>\glswiftouter</td>
<td>29, 30, 796, 913</td>
</tr>
<tr>
<td>\glswiftsymbol</td>
<td>796, 797</td>
</tr>
<tr>
<td>\glstartrange</td>
<td>§5.8; 262, 263, 733, 797, 905</td>
</tr>
<tr>
<td>\glssubentryitem</td>
<td>420, 797</td>
</tr>
<tr>
<td>\GLSsymbol</td>
<td>797</td>
</tr>
<tr>
<td>\Glssymbol</td>
<td>797</td>
</tr>
<tr>
<td>\glssymbol</td>
<td>368, 369, 636, 797, 798, 953, 954</td>
</tr>
<tr>
<td>\glssymbolaccessdisplay</td>
<td>421, 504, 511, 798</td>
</tr>
<tr>
<td>\glssymbolplural</td>
<td>637, 798</td>
</tr>
<tr>
<td>\glssymbolpluralaccessdisplay</td>
<td>504, 512, 798</td>
</tr>
<tr>
<td>\glssymbolsgroupname</td>
<td>432, 798, 993</td>
</tr>
<tr>
<td>\glstableblocksentry</td>
<td>488, 798</td>
</tr>
<tr>
<td>\glstableblocksentrysep</td>
<td>§8.7.4.1; 488, 799</td>
</tr>
<tr>
<td>\glstableblockwidth</td>
<td>§8.7.4.4; 496, 799</td>
</tr>
<tr>
<td>\glstablecaption</td>
<td>§8.7.4.2; 489, 799</td>
</tr>
<tr>
<td>\glstablecenteralign</td>
<td>§8.7.4.4; 495, 799</td>
</tr>
<tr>
<td>\glstableChildEntries</td>
<td>§8.7.4.1; 487, 799, 800, 803</td>
</tr>
<tr>
<td>\glstabledesccolalign</td>
<td>§8.7.4.4; 495, 799</td>
</tr>
<tr>
<td>\glstableDescFmt</td>
<td>§8.7.4.4; 497, 498, 799</td>
</tr>
<tr>
<td>\glstablechild</td>
<td>§§8.7.4.2, 8.7.4.4; 487, 799, 800</td>
</tr>
<tr>
<td>\glstabledesccolalign</td>
<td>§8.7.4.4; 497, 799</td>
</tr>
<tr>
<td>\glstabledescheader</td>
<td>§§8.7.4.2, 8.7.4.4; 498, 799</td>
</tr>
<tr>
<td>\glstableDescFmt</td>
<td>§8.7.4.4; 497, 498, 799</td>
</tr>
<tr>
<td>\glstabledescheader</td>
<td>§§8.7.4.2, 8.7.4.4; 498, 799</td>
</tr>
<tr>
<td>\glstabledescwidth</td>
<td>§8.7.4.4; 496, 800</td>
</tr>
<tr>
<td>\glstableHeaderFmt</td>
<td>§8.7.4.4; 499, 800</td>
</tr>
<tr>
<td>\glstableiffilter</td>
<td>§8.7.4.4; 484, 487, 800</td>
</tr>
<tr>
<td>\glstableiffilterchild</td>
<td>§8.7.4.1; 487, 800</td>
</tr>
<tr>
<td>\glstableifhasotherfield</td>
<td>§8.7.4.4; 498, 800</td>
</tr>
<tr>
<td>\glstableleftalign</td>
<td>§8.7.4.4; 494, 800</td>
</tr>
<tr>
<td>\glstablenamecolalign</td>
<td>§8.7.4.4; 495, 801</td>
</tr>
<tr>
<td>\glstableNameFmt</td>
<td>§8.7.4.4; 484, 497, 801</td>
</tr>
<tr>
<td>\glstablenameheader</td>
<td>§§8.7.4.2, 8.7.4.4; 490, 499, 801</td>
</tr>
<tr>
<td>\glstablenamewidth</td>
<td>§8.7.4.4; 496, 801</td>
</tr>
<tr>
<td>\glstablenewline</td>
<td>§8.7.4.4; 488, 494, 801</td>
</tr>
<tr>
<td>\glstablenextcaption</td>
<td>§§8.7.4.2, 8.7.4.4; 489, 801, 802</td>
</tr>
<tr>
<td>\glstableother</td>
<td>§8.7.4.4; 498, 801</td>
</tr>
<tr>
<td>\glstableothercolalign</td>
<td>§8.7.4.4; 496, 802</td>
</tr>
<tr>
<td>\glstableotherfield</td>
<td>§8.7.4.4; 498, 802</td>
</tr>
<tr>
<td>\glstableOtherFmt</td>
<td>§8.7.4.4; 498, 802</td>
</tr>
</tbody>
</table>
Index

\glstableotherheader §§8.7.4.2, 8.7.4.4; 490, 499, 802
\glstableotherwidth §§8.7.4.4; 496, 802
\glstablepostnextcaption §8.7.4.2; 489, 802
\glstablepostpreambleskip §8.7.4.4; 496, 802
\glstablePreChildren §8.7.4.1; 487, 802
\glstableprepostpreambleskip §8.7.4.4; 497, 803
\glablerightalign §8.7.4.4; 495, 803
\glablessetstyle §8.7.4.3; 491, 803
\glablesSubDescFmt §8.7.4.4; 498, 803
\glablesSubentries environment §8.7.4.1; 487, 803, 983
\glablesSubentryalign §8.7.4.1; 488, 803
\glablesSubNameFmt §8.7.4.4; 497, 803
\glablesSubOtherFmt §8.7.4.4; 498, 803
\glablesSymbolFormat §8.7.4.4; 495, 804
\glablesSymbolFmt §8.7.4.4; 497, 804
\glablesSymbolHeader §§8.7.4.2, 8.7.4.4; 490, 499, 804
\glablesSymbolWidth §§8.7.4.4; 496, 804
\glarget . 29, 257, 383, 420, 527, 804, 918
\GLText . . 204, 212, 725, 727, 804
\GLtext . . 212, 725, 727, 805
\gltext . 5, 23, 37, 43, 47, 51, 81, 183, 185, 192, 202, 204, 212, 235, 241, 245, 253, 285, 322, 367, 520, 536, 626, 725, 727, 805, 950, 953
\gltext-like . . 626
\gltextaccessdisplay . . 503, 510, 805
\gltextformat . . 190, 191, 239, 240, 255, 524, 642, 644, 805, 824
\gltextup . . 157, 805, 902
\gltextchar . . 527, 805
\gltopicAssignSubIndent §8.7.3; 481, 806
\gltopicAssignWidest §8.7.3; 481, 806
\gltopicCols §§8.7.3; 477, 806
\gltopicColsEnv . . §8.7.3; 476, 806
\gltopicDesc §8.7.3; 481, 806
\gltopicGroupHeading §8.7.3; 479, 480, 806
\gltopicInit §8.7.3; 479, 806
\gltopicItem §8.7.3; 481, 807
\gltopicLoc §8.7.3; 481, 807
\gltopicMarker §8.7.3; 480, 807
\gltopicMidSkip §8.7.3; 480, 807
\gltopicParIndent §8.7.3; 478, 807
\gltopicPostSkip §8.7.3; 481, 807
\gltopicPreSkip §8.7.3; 480, 807
\gltopicSubGroupHeading §8.7.3; 480, 807
\gltopicSubIndent §8.7.3; 479, 808
\gltopicSubItem §8.7.3; 481, 808
\gltopicSubItemBox §8.7.3; 482, 808
\gltopicSubItemParIndent §8.7.3; 479, 808
\gltopicSubItemSep §8.7.3; 482, 808
\gltopicSubLoc §8.7.3; 482, 808
\gltopicSubNameFont §8.7.3; 481, 808
\gltopicSubPreLocSep §8.7.3; 482, 809
\gltopicTitle §8.7.3; 480, 809
\gltopicTitleFont §8.7.3; 480, 809
\glstreechilddesc §8.6.5.4; 438, 809
\glstreeChildDescLoc §8.6.5.4; 439, 809
\glstreechildpredesc §8.6.5.4; 437, 809
\glstreechildprelocation §8.6.5.4; 437, 439, 809
\glstreechildsymbol §8.6.5.4; 438, 809
\glstreedefaultnamefmt §8.6.5.4; 436, 810
\glstree desc . . §8.6.5.4; 438, 810
\glstreeDescLoc §8.6.5.4; 439, 809
\glstreeGroupHeading §8.6.5.4; 435, 449, 810
\glstreeGroupHeading . . §8.6.5.4; 436, 810
\glstreeGroupHeading . . §8.6.5.4; 436, 810
\glstreeGroupHeading . . §8.6.5.4; 436, 810
\glstreeItem . . . . . . . 810
\glstreeNamefmt . . 436, 449, 811
\glstreenavfmt §8.6.5.4; 436, 811

1019
## Index

<table>
<thead>
<tr>
<th>Command/Usage</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>\gls-tree-no-desc-symbol-pre-location</td>
<td>§8.6.5.4; 439, 811</td>
</tr>
<tr>
<td>\gls-tree-no-name-child-desc</td>
<td>§8.6.5.4; 438, 811</td>
</tr>
<tr>
<td>\gls-tree-no-name-desc-loc</td>
<td>811</td>
</tr>
<tr>
<td>\gls-tree-no-name-desc</td>
<td>§8.6.5.4; 438, 811</td>
</tr>
<tr>
<td>\gls-tree-no-name-symbol</td>
<td>§8.6.5.4; 438, 811</td>
</tr>
<tr>
<td>\gls-tree-pre-desc</td>
<td>§8.6.5.4; 437, 812</td>
</tr>
<tr>
<td>\gls-tree-pre-header</td>
<td>§8.6.5.4; 436, 812</td>
</tr>
<tr>
<td>\gls-tree-relocation</td>
<td>§8.6.5.4; 437, 439, 812</td>
</tr>
<tr>
<td>\gls-tree-subgroup-item</td>
<td>812</td>
</tr>
<tr>
<td>\gls-tree-sub-item</td>
<td>448, 812</td>
</tr>
<tr>
<td>\gls-tree-sub-pre-header</td>
<td>812</td>
</tr>
<tr>
<td>\gls-tree-sub-sub-item</td>
<td>448, 812</td>
</tr>
<tr>
<td>\gls-tree-symbol</td>
<td>§8.6.5.4; 438, 813</td>
</tr>
<tr>
<td>\gls-trigger-record-format</td>
<td>§11.4; 550, 582, 813, see also \rgls &amp; --record-count</td>
</tr>
<tr>
<td>\gls-unexpanded-field-value</td>
<td>813</td>
</tr>
<tr>
<td>\glsunset</td>
<td>283, 286–289, 310, 643, 813, 836, 897, see also \gls-local-unset, \gls-reset, if \gls-used &amp; \GlsBrIfUnusedOrUndefined</td>
</tr>
<tr>
<td>\glsunset-all-283, 813, see also \glsunset, \gls-reset-all, if \gls-used &amp; \GlsBrIfUnusedOrUndefined</td>
<td></td>
</tr>
<tr>
<td>\glsunset-category-attribute</td>
<td>§10.2.2; 529, 813</td>
</tr>
<tr>
<td>\gls-update-widest</td>
<td>§8.6.5.4; 440, 700, 702, 813</td>
</tr>
<tr>
<td>\gls-upper-case</td>
<td>§5.2.3; 200, 814, 942</td>
</tr>
<tr>
<td>\gls-use-abbrv-font</td>
<td>§4.5.2; 159, 814</td>
</tr>
<tr>
<td>\gls-use-long-font</td>
<td>§4.5.2; 159, 814</td>
</tr>
<tr>
<td>\gls-user-description</td>
<td>88, 100, 101, 137, 814</td>
</tr>
<tr>
<td>\GLS-useri</td>
<td>814</td>
</tr>
<tr>
<td>\GLSuseri</td>
<td>814</td>
</tr>
<tr>
<td>\GLSuseri</td>
<td>637, 814</td>
</tr>
<tr>
<td>\gls-user-access-display</td>
<td>507, 514, 815</td>
</tr>
<tr>
<td>\GLS-userii</td>
<td>815</td>
</tr>
<tr>
<td>\GLSuserii</td>
<td>815</td>
</tr>
<tr>
<td>\gls-userii</td>
<td>638, 815</td>
</tr>
<tr>
<td>\gls-user-access-display-200</td>
<td>507, 515, 815</td>
</tr>
<tr>
<td>\GLS-userii</td>
<td>815</td>
</tr>
<tr>
<td>\gls-user-access-display-201</td>
<td>638, 816</td>
</tr>
<tr>
<td>\gls-user-access-display-202</td>
<td>508, 515, 816</td>
</tr>
<tr>
<td>\GLS-userii</td>
<td>816</td>
</tr>
<tr>
<td>\gls-user-access-display-203</td>
<td>638, 816</td>
</tr>
<tr>
<td>\gls-user-access-display-204</td>
<td>508, 516, 817</td>
</tr>
<tr>
<td>\GLS-useriv</td>
<td>817</td>
</tr>
<tr>
<td>\gls-useriv</td>
<td>817</td>
</tr>
<tr>
<td>\gls-useriv</td>
<td>638, 817</td>
</tr>
<tr>
<td>\gls-user-access-display-205</td>
<td>509, 516, 817</td>
</tr>
<tr>
<td>\GLS-useriv</td>
<td>817</td>
</tr>
<tr>
<td>\gls-useriv</td>
<td>818</td>
</tr>
<tr>
<td>\gls-useriv</td>
<td>639, 818</td>
</tr>
<tr>
<td>\gls-user-access-display-206</td>
<td>509, 516, 818</td>
</tr>
<tr>
<td>\GLS-useriv</td>
<td>818</td>
</tr>
<tr>
<td>\gls-useriv</td>
<td>818</td>
</tr>
<tr>
<td>\gls-user-access-display-207</td>
<td>639, 818</td>
</tr>
<tr>
<td>\gls-user-access-display-208</td>
<td>509, 516, 818</td>
</tr>
<tr>
<td>\GLS-useriv</td>
<td>818</td>
</tr>
<tr>
<td>\gls-user-access-display-209</td>
<td>819, 831, 839, 892</td>
</tr>
<tr>
<td>\gls-user-access-display-210</td>
<td>§4.5.3.1; 168, 818</td>
</tr>
<tr>
<td>\gls-user-access-display-211</td>
<td>§4.5.3.1; 168, 818</td>
</tr>
<tr>
<td>\gls-user-access-display-212</td>
<td>§4.5.3.1; 168, 818</td>
</tr>
<tr>
<td>\Glsxtr</td>
<td>§13; 605, 606, 819</td>
</tr>
<tr>
<td>\Glsxtr</td>
<td>§13; 604–606, 819, 831, 839, 892</td>
</tr>
<tr>
<td>\Glsxtr-reset &amp; if \gls-used &amp; \GlsBrIfUnusedOrUndefined</td>
<td>§8.4.3.2; 412, 414, 819</td>
</tr>
<tr>
<td>\Glsxtr-record &amp; if \gls-used &amp; \GlsBrIfUnusedOrUndefined</td>
<td>§11.5.6; 581, 819</td>
</tr>
<tr>
<td>\Glsxtr-record &amp; if \gls-used &amp; \GlsBrIfUnusedOrUndefined &amp; if \gls-used &amp; \GlsBrIfUnusedOrUndefined</td>
<td>§11.5.6; 581, 582, 819</td>
</tr>
<tr>
<td>\Glsxtr-resource &amp; if \gls-used &amp; \GlsBrIfUnusedOrUndefined</td>
<td>§11; 17, 536, 819</td>
</tr>
<tr>
<td>\Glsxtr-abbreviation-font</td>
<td>§5.5.2; 61, 65, 237, 241, 256, 734, 820</td>
</tr>
<tr>
<td>\Glsxtr-abbreviation-font &amp; footnote</td>
<td>145, 820</td>
</tr>
<tr>
<td>\Glsxtr-abbreviation-font &amp; plural-suffix &amp; §4.5.3.1; 169, 820</td>
<td>§4.5.3.1; 169, 820</td>
</tr>
<tr>
<td>\Glsxtr-abbreviation-font &amp; plural-suffix &amp; §4.5.3.1; 169, 821</td>
<td>§4.5.3.1; 169, 821</td>
</tr>
<tr>
<td>\Glsxtr-abbreviation-font &amp; plural-suffix &amp; §4.5.3.1; 169, 822</td>
<td>§4.5.3.1; 169, 822</td>
</tr>
<tr>
<td>\Glsxtr-abbreviation-font &amp; plural-suffix &amp; §4.5.3.1; 169, 823</td>
<td>§4.5.3.1; 169, 823</td>
</tr>
<tr>
<td>Field</td>
<td>Reference</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>\glsxtrbookindexlastmarkfmt</td>
<td>§8.7.1; 449, 829</td>
</tr>
<tr>
<td>\glsxtrbookindexlocation</td>
<td>§8.7.1; 449, 829</td>
</tr>
<tr>
<td>\glsxtrbookindexmarkentry</td>
<td>§8.7.1; 449, 829</td>
</tr>
<tr>
<td>\glsxtrbookindexmulticolsenv</td>
<td>§8.7.1; 444, 830</td>
</tr>
<tr>
<td>\glsxtrbookindexname</td>
<td>§8.7.1; 445, 449, 450, 830</td>
</tr>
<tr>
<td>\glsxtrbookindexparentchildsep</td>
<td>§8.7.1; 446, 830</td>
</tr>
<tr>
<td>\glsxtrbookindexparentsseparator</td>
<td>§8.7.1; 446, 830</td>
</tr>
<tr>
<td>\glsxtrbookindexpregroupskip</td>
<td>§8.7.1; 449, 830</td>
</tr>
<tr>
<td>\glsxtrbookindexprelocation</td>
<td>§8.7.1; 445, 446, 830</td>
</tr>
<tr>
<td>\glsxtrbookindexsubatendgroup</td>
<td>§8.7.1; 447, 830</td>
</tr>
<tr>
<td>\glsxtrbookindexsubbetween</td>
<td>§8.7.1; 447, 830</td>
</tr>
<tr>
<td>\glsxtrbookindexsublocation</td>
<td>§8.7.1; 446, 831</td>
</tr>
<tr>
<td>\glsxtrbookindexsubname</td>
<td>§8.7.1; 445, 450, 831</td>
</tr>
<tr>
<td>\glsxtrbookindexsubprelocation</td>
<td>§8.7.1; 446, 831</td>
</tr>
<tr>
<td>\glsxtrbookindexsubsubatendgroup</td>
<td>§8.7.1; 447, 831</td>
</tr>
<tr>
<td>\glsxtrbookindexsubsubbetween</td>
<td>§8.7.1; 447, 831</td>
</tr>
<tr>
<td>\glsxtrcat</td>
<td>§13; 606, 831</td>
</tr>
<tr>
<td>\glsxtr{category}(field)accsupp</td>
<td>742, 831, 898</td>
</tr>
<tr>
<td>\glsxtrchecknohyperfirst</td>
<td>§5.1.1; 186, 189, 523, 832</td>
</tr>
<tr>
<td>\glsxtrclearlabelprefixes</td>
<td>§11.5.7; 589, 832, see also \glsxtraddlabelprefix &amp; \glsxtrprependlabelprefix</td>
</tr>
<tr>
<td>\GlsXtrClearUnsetBuffer</td>
<td>§5.10.1; 288, 289, 832</td>
</tr>
<tr>
<td>\glsxtrcombiningdiacriticIIrules</td>
<td>569, 832</td>
</tr>
</tbody>
</table>

1022
Index

\GlsXtrDiscardUnsetBuffering §5.10.1; 288, 836, see also
\GlsXtrStartUnsetBuffering & \GlsXtrStopUnsetBuffering
\glsextrdisplayendloc ...... §8.6.3; 430, 836, 837
\glsextrdisplayendlochook ...... §8.6.3; 430, 836
\glsextrdisplaylocnameref §11.5.6; 582, 584, 821, 833, 837, 840, 870, 886, 899, 904, 918, 934
\glsextrdisplaysingleloc ...... §8.6.3; 430, 837
\glsextrdisplaystartloc ...... §8.6.3; 430, 837, 871
\glsextrdisplaysupploc ...... §11.5.5; 581, 837, 885
\glsextrdautoindexname ...... §12; 268, 600, 602, 837
\glsextrdropostpunc ...... §5.5.4; 249, 837
\GlsXtrDowrglossaryhook §5.8; 267, 837
\GlsXtrDualBackLink §11.5.7; 587, 838
\GlsXtrDualField §11.5.7; 588, 838
\glsextrdeffield ...... §3.5; 38, 838
\glsextermrevert ...... 158, 838
\glsextrarraysuffix .... 71–73, 78–80, 85–87, 97–99, 126–129, 158, 838
\GlsXtrEnableEntryCounting §6.1; 311, 312, 838
\GlsXtrEnableEntryUnitCounting §6.1; 316, 319, 838
\GlsXtrEnableIndexFormatOverride §12; 602, 839
\GlsXtrEnableInitialTagging §4.4; 56, 58, 523, 839, 917
\GlsXtrEnableLinkCounting §6.2; 320, 322, 323, 527, 839
\GlsXtrEnableInTheFly §13; 604, 616, 819, 839, 915
\GlsXtrEnablePreLocationTag §8.6.3; 429, 430, 839
\glsextrenablerecordcount §11.4; 56, 552, 839
\glsextrforesetfor §5.13; 304, 840
\glsextrentryfmt §5.12.2; 301, 840
\glsextrentryfmt ...... §5.12.2; 299, 302, 840
\glsextrentryparentname §5.11; 294, 840
\glsextrequationlocloc §11.5.6; 584, 840
\GlsXtrExpandedFmt §5.5; 237, 292, 840
\glsextrfielddolistloop §15.4; 39, 307, 840
\glsextrfieldforlistloop §15.4; 39, 307, 841
\glsextrfieldformatcsvlist §5.13; 39, 305, 841
\glsextrfieldformatlist §15.4; 307, 841
\glsextrfieldifixinlist §5.14; 307, 841
\glsextrfieldlistadd §3.5; 39, 807, 841
\glsextrfieldlistgadd §3.5; 39, 841
\glsextrfieldlistxadd §3.5; 39, 842
\glsextrfieldtitlecase §5.11; 294, 842
\glsextrfieldtitlecasecs §5.11; 294, 525, 842
\glsextrfieldxfimain §5.14; 308, 414, 842
\glsextrfieldxfiman §5.14; 308, 414, 842
\glsextrfirstscfont §5.12.2; 301, 842
\glsextrfirstsmfont §5.12.2; 186–190, 298, 299, 301, 843
\GlsXtrFmt §5.12.2; 186–190, 298, 299, 301, 302, 614, 843, 844
\GlsXtrFmtDefaultOptions §5.1.1; 186, 188, 299, 843
\glsextrfmt §5.12.2; 301, 843
\glsextrfmt §5.12.2; 301, 843
\GlsXtrFmtDefaultOptions §5.1.1; 186, 188, 299, 843
\glsextrfmt §5.12.2; 299, 843
\glsextrfmt §5.12.2; 299, 843
\glsextrfmt §5.12.2; 298, 840, 843
\glsextrfmt §5.12.2; 298, 840, 843
\glsextrfmt §5.12.2; 840, 843
\glsextrfmt §5.12.2; 840, 843
\glsextrfmt §5.12.2; 840, 843
\glsextrfootnotedescname §11.5.6; 585, 844
\glsextrfootnotedescsort §11.5.6; 585, 844
\glsextrfootnotelongformat §11.5.6; 585, 844
\glsextrfootnotelongplformat §11.5.6; 585, 844
\glsextrfootnotename §11.5.6; 585, 844
\glsextrforcsvfield §5.13; 39, 304, 1023
Index

840, 844  
\GlsXtrForeignText \S5.12.1; 296, 297, 845  
\GlsXtrForeignTextField \S5.12.1; 296, 845  
\GlsXtrFormatLocationList \S8.6.3; 429, 845  
\GlsXtrForUnsetBufferedList \S5.10.1; 288, 293, 845  
glsxtrfractionrules \S5.12.1; 296, 845  
glsxtrformat \S4.3; Table 4.1; 429, 479, 845, 846  
glsxtrfull \S5.10.1; 288, 293, 845  
glsxtrfullformat \S4.3; Table 4.1; 43, 47, 50, 51, 53, 54, 105, 139–143, 159, 161, 175, 185, 211, 236, 253, 255, 285, 368, 625, 627, 657, 658, 667, 683, 688, 745, 845, 846, 847, 865, 901, 907, 946  
glsxtrfullformat \S8.6.3; 432, 479, 851  
glsxtrfullformatpl \S4.3; Table 4.1; 43, 47, 50, 51, 53, 54, 105, 139–143, 159, 161, 175, 185, 211, 236, 253, 255, 285, 368, 625, 627, 657, 658, 667, 683, 688, 745, 845, 846, 847, 865  
glsxtrfullfullformat \S5.3.3; 226, 851, see also \Glsfmtfirst & \Glsxtrtitlefirst  
glsxtrfullfullformatpl \S5.3.3; 226, 851, see also \Glsfmtfirst & \Glsxtrtitlefirst  
glsxtrfullfunction \S4.3; Table 4.1; 54, 140, 141, 175, 211, 627, 683, 689, 749, 846, 847, 866  
glsxtrfullfulllength \S5.3.3; 226, 851, see also \Glsfmtfirst & \Glsxtrtitlefirst  
glsxtrfullfulllengthpl \S5.3.3; 226, 851, see also \Glsfmtfirst & \Glsxtrtitlefirst  
glsxtrfullfunctionpl \S4.3; Table 4.1; 54, 175, 211, 683, 689, 748, 845, 846  
glsxtrfullfunctionpl \S8.6.3; 432, 479, 851  
glsxtrfullfullformatpl \S5.3.3; 226, 851, see also \Glsfmtfirst & \Glsxtrtitlefirst  
glsxtrfullfullformatpl \S5.3.3; 226, 851, see also \Glsfmtfirst & \Glsxtrtitlefirst  
glsxtrfullfunctio...
see also \glsfmtfirstpl & \glstitlefirstplural
\GLSxtrheadfull $\S 5.3.3$; 222, 852, see also
\glsfmtfull & \glstitlefull
\GLsxheadtext $\S 5.3.3$; 222, 856, see also
\glstitletext
\GLsxheadlong $\S 5.3.3$; 220, 853
\GLsxthiernamesep $\S 5.11$; 295, 296, 857
\glsXtrheadfullpl $\S 5.3.3$; 223, 853, see also \glsfmtfullpl & \glstitlefullpl
Index

\glxstrifhasfield \ §5.15; 280, 281, 295, 304–309, 580, 841, 845, 858, 859, 862
\GlsXtrIfHasNonZeroChildCount \ §11.5.4; 580, 860
\glxstrifheaduc \ §5.3.3; 216, 860
\glxstrifhyphenstart \ §4.5.2; 149, 160, 860
\glxstrifindexing \ §5.8; 268, 860
\GlsXtrIfInGlossary \ §8; 379, 860, see also \ifglossaryexists & \glxstrifemptyglossary
\glxstrifinlabelprefixlist \ §11.5.7; 589, 861, see also \glxstraddlabelprefix & \glxstrprependlabelprefix
\glxstrifinmark \ §5.3.3; 214, 215, 227, 681, 861
\glxstrifintoc \ §5.3.3; 214, 217, 861
\glxstrifkeydefined \ §3.2; 35, 861
\glxstriflabelinlist \ §8.4.3; 408, 861
\GlsXtrIfLinkCounterDef \ §6.2; 321, 861
\glxstrimulti \ §7.13; 373, 861
\glxstrinextpunc \ §5.5.4; 249, 822, 837, 862, 907
\glxstriperiod \ §5.5.4; 248, 249, 836, 862
\glxstrifrecordtrigger \ §11.4; 549, 862, 900, see also \rgls, \GlsXtr-SetRecordCounterAttribute & --record-count
\GlsXtrIfUnusedOrUndefined \ §5.10; 15, 268, 286, 862, see also \ifglsused & \glxstrifwasfirstuse
\GlsXtrIfValueInFieldCsvList \ §5.13; 306, 307, 862, 982
\glxstrifwasfirstuse \ §5.10; 52, 54, 192, 247, 253, 254, 285, 286, 353, 354, 863, 906
\glxstrifwasglslike \ §5.5.4; 253, 254, 863
\glxstrifwasglslikeandfirstuse \ §5.5.4; 253, 286, 863
\glxstrifwassubsequentshort \ §5.5.4; 254, 863
\glxstrifwassubsequentuse \ §5.5.4; 253, 863
\GlsXtrIfXpFieldEqXpStr \ §5.15; 309, 863
\glxstrIgnorableRules \ 572, 864
\glxstrinlinkcounter \ §6.2; 320, 864
\glxstrindexaliased \ §5.9.3; 281, 282, 864
\GlsXtrIndexCounterLink \ §11.5.8; 599, 864
\glxstrindexseealso \ §5.9.3; 22, 33, 270, 274, 281, 864, see also \glssee
\glxstrinithyperoutside \ §5.1.1; 186, 190, 864
\glxstrinitwrgyzloss \ §5.1.1; 186, 189, 194, 267, 865
\glxstrinitwrgyzlossbeforefalse \ 189, 865
\glxstrinitwrgyzlossbeforetrue \ 189, 865
\GLSxtrlinefullformat \ 175, 865
\GLSxtrlinefullformat \ 175, 865
\GLSxtrlinefullformat \ 175, 865
\GLSxtrlinefullpformat \ 175, 865
\GLSxtrlinefullpformat \ 175, 866
\GLSxtrlinefullpformat \ 175, 866
\glxstrinsertinsidefalse \ 131, 866
\glxstrinsertinsidetrue \ 131, 866
\GlsXtrInternalLocationHyperlink \ 866
\glxstrLatinA \ 573, 575, 866
\glxstrLatinAA \ 577, 866
\glxstrLatinAELegature \ 576, 867
\glxstrLatinE \ 575, 867
\glxstrLatinEszettsSs \ 576, 867
\glxstrLatinEszettsSz \ 576, 867
\glxstrLatinEth \ 576, 867
\glxstrLatinH \ 575, 867
\glxstrLatinI \ 575, 867
\glxstrLatinInsularG \ 577, 867
\glxstrLatinK \ 575, 868
\glxstrLatinL \ 868
\glxstrLatinLslash \ 577, 868
\glxstrLatinM \ 575, 868
\glxstrLatinN \ 575, 868
\glxstrLatinO \ 575, 868
\glxstrLatinOLEgature \ 577, 868
\glxstrLatinOslash \ 577, 868
\glxstrLatinP \ 575, 869
\glxstrLatinS \ 576, 869

1026
 Index

\glshtrMathItalicAlpha  \ldots  579, 877  \par \GLSxtrmultientryadjustedname \S7.14; 375, 882, 883, 884
\glshtrMathItalicBeta  \ldots  579, 877
\glshtrMathItalicChi  \ldots  579, 878
\glshtrMathItalicDelta  \ldots  579, 878
\glshtrMathItalicDigamma  \ldots  579, 878
\glshtrMathItalicEpsilon  \ldots  579, 878
\glshtrMathItalicEta  \ldots  579, 878
\glshtrMathItalicGamma  \ldots  579, 878
\glshtrMathItalicGreekIIrules  \ldots  578, 878
\glshtrMathItalicGreekIrules  \ldots  578, 879
\glshtrMathItalicIota  \ldots  579, 879
\glshtrMathItalicKappa  \ldots  579, 879
\glshtrMathItalicLambda  \ldots  579, 879
\glshtrMathItalicLowerGreekIIrules  \ldots  579, 879
\glshtrMathItalicLowerGreekIrules  \ldots  579, 879
\glshtrMathItalicMu  \ldots  579, 879
\glshtrMathItalicNabla  \ldots  579, 879
\glshtrMathItalicNu  \ldots  579, 880
\glshtrMathItalicOmega  \ldots  579, 880
\glshtrMathItalicOmicron  \ldots  579, 880
\glshtrMathItalicPartial  \ldots  579, 880
\glshtrMathItalicPhi  \ldots  579, 880
\glshtrMathItalicPi  \ldots  579, 880
\glshtrMathItalicPsi  \ldots  579, 880
\glshtrMathItalicRho  \ldots  579, 880
\glshtrMathItalicSigma  \ldots  579, 881
\glshtrMathItalicTau  \ldots  579, 881
\glshtrMathItalicTheta  \ldots  579, 881
\glshtrMathItalicUpperGreekIIrules  \ldots  578, 881
\glshtrMathItalicUpperGreekIrules  \ldots  578, 881
\glshtrMathItalicUpperGreekIrules  \ldots  578, 881
\glshtrMathItalicUpsilon  \ldots  578, 882
\glshtrMathItalicXi  \ldots  578, 882
\glshtrMathUpGreekIIrules  \ldots  578, 882
\glshtrMathUpGreekIrules  \ldots  578, 882
\glshtrMUFUsave  \S11; 537, 882
\GlsXtrMglsOrGls  \S7; 328, 882
\glshtrmglsWarnAllSkipped  \S7.9.5; 361, 882
\GLSxtrmultientryadjustedname  \S7.14; 376, 883
\GLSxtrmultientryadjustednamefmt  \S7.14; 376, 883
\GLSxtrmultientryadjustednameother  \S7.14; 376, 884
\GLSxtrmultientryadjustednameotherpostsep  \S7.14; 376, 884
\GLSxtrmultientryadjustednamepresep  \S7.14; 376, 884
\GLSxtrmultientryadjustednameotherpostsep  \S7.14; 376, 884
\GLSxtrmultientryadjustednameotherpresep  \S7.14; 376, 884
\GLSxtrmultilastotherindex  \S7.13; 374, 885
\GLSxtrmultilist  \S7.13; 373, 885
\GLSxtrmultimain  \S7.13; 373, 885
\GLSxtrmultimainindex  \S7.13; 374, 885
\GLSxtrmultisupplocation  \S11.5.5; 581, 885
\GLSxtrmultitotalelements  \S7.13; 374, 885
\GLSxtrnamelomlink  \S11.5.6; 586, 885
\GLSxtrnamereflink  585, 843, 844, 886
\glshtrnewabbrevpresetkeyhook  \S4.1.5; 45, 162, 886
\glshtrnewgls  \S5.7; 258, 260, 589, 594,
Index

\glsxtrsconlysuffix 113, 155, 902
\glsxstrscrevert 137, 155, 157, 902, 903
\glsxtrscssuffix 67–69, 75, 76, 82, 83, 94, 95, 118–121, 137, 155, 157, 171, 682, 903
\glsxtrscuserrevert 136, 137, 903
\glsxtrscusersuffix 91, 137, 903
\glsxtrseealsolabels §5.9.2; 280, 903
\glsxtrseelist §5.13; 302, 903
\glsxtrseelists §5.9.2; 279, 903, 904, see also \glsxtrseelistsencap & \glsxtrseelistsdelim
\glsxtrseelistsdelim §5.9.2; 280, 903
\glsxtrsetactualanchor §11.5.6; 583, 904
\GlsXtrSetActualChar §12; 603, 904
\glsxtrsetaliasnoindex §5.9.3; 271, 281, 282, 864, 904
\GlsXtrSetAltModifier §5; 184, 312, 329, 882, 904
\glsxtrsetbibglsaux §2.4; 26, 904
\glsxtrsetcategory §10; 517, 904
\glsxtrsetcategoryforall §10; 517, 905
\glsxtrsetcomplexstyle §4.5.3.1; 165, 166, 905
\GlsXtrSetDefaultGlsOpts §5.1.1; 186–190, 281, 905
\GlsXtrSetDefaultNumberFormat §5.1.1; 188, 262, 905
\GlsXtrSetDefaultRangeFormat §5.8; 262, 797, 905
\GlsXtrSetEncapChar §12; 603, 905
\GlsXtrSetEscChar §12; 603, 906
\GlsXtrSetField §3.5; 37–40, 275, 394, 517, 625, 700, 702, 835, 869, 906, 982
\glsxtrsetfieldifexists §3.5; 39, 906
\glsxtrsetgrouptitle §8.6.4; 34, 393, 432, 633, 906
\glsxtrsetglossarylabel §8.3; 382, 906
\GlsXtrSetLevelChar §12; 603, 906
\glsxtrsetlongfirstuse §4.3; 52, 906
\GlsXtrSetPlusModifier §5; 184, 907
\GlsXtrSetRecordCountAttribute §11.4; 549, 907
\GlsXtrSetStarModifier §5; 184, 907
\glsxtrsetupfulldefs §4.3; 54, 907
\glsxtrsetwidest §11.5.3; 579, 908
\glsxtrsetwidestfallback §11.5.3; 580, 908
\GlsXtrshort §4.3; Table 4.1; 52, 204, 207, 218, 687, 690, 805, 908
\GlsXtrshort §4.3; Table 4.1; 51, 207, 687, 690, 805, 908
\GlsXtrshort §4.3; Table 4.1; 18, 19, 43, 47, 51, 54, 55, 61, 63, 66, 74, 82, 93, 103, 115, 159, 177, 183, 185, 202, 204, 207, 236, 241, 248, 253, 337, 367, 657, 658, 687, 690, 805, 908, 911, 953, 966
\GlsXtrshortdescname 66, 147, 908
\GlsXtrshortformat 177, 908
\GlsXtrshortformat 177, 909
\GlsXtrshortformat 114, 177, 178, 180, 876, 908, 909, 910, 912
\GlsXtrshortformatgrp 178, 909
\GlsXtrshortformatgrp 178, 909
\GlsXtrshortformatgrp 178, 909
\glsxtrshorthyphen 152, 909
\GLXtrshorthyphenlong 152, 909
\glsxtrshorthyphenlong 152, 909, 910
\glsxtrshorthyphenlongsort 108, 148, 910
\GlsXtrshortlongdescname 94, 110, 134, 910
\glsxtrshortlongdescsort 94, 100, 110, 134, 910
\GLXtrshortlongformat 180, 910
\GLXtrshortlongformat 180, 910
\glsxtrshortlongformat 65, 93, 114, 180, 910, 911
\glsxtrshortlongname 108, 134, 911
1031
Index

\Glsxtrtitleshort §5.3.3; 217, 218, 922
\Glsxtrtitlefull §5.3.3; 222, 751, 919, see also \Glsfmtfull
\Glsxtrtitlefull §5.3.3; 222, 751, 919, see also \Glsfmtfull & \Glsxtrheadfull
\Glsxtrtitlefull §5.3.3; 221, 751, 919, see also \Glsfmtfull & \Glsxtrheadfull
\Glsxtrtitlefullpl §5.3.3; 223, 751, 919, see also \Glsfmtfullpl & \Glsxtrheadfull
\Glsxtrtitlefullpl §5.3.3; 222, 752, 920, see also \Glsfmtfullpl & \Glsxtrheadfull
\Glsxtrtitlefullpl §5.3.3; 222, 752, 920, see also \Glsfmtfullpl & \Glsxtrheadfull
\Glsxtrtitletext §5.3.3
\Glsxtrtitlelongpl §5.3.3; 223, 751, 919, see also \Glsfmtlongpl
\Glsxtrtitlelongpl §5.3.3; 223, 752, 920, see also \Glsfmtlongpl & \Glsxtrheadlongpl
\Glsxtrtitlename §5.3.3; 223, 921, see also \Glsfmtname & \Glsxtrheadname
\Glsxtrtitlename §5.3.3; 223, 921, see also \Glsfmtname & \Glsxtrheadname
\Glsxtrtitlename §5.3.3; 223, 921, see also \Glsfmtname & \Glsxtrheadname
\Glsxtrtitleopts §5.3.2; 206, 216, 217, 921
\Glsxtrtitleorpdforheading §5.3.3; 215, 421, 921
\Glsxtrtitleplural §5.3.3; 225, 921, see also \Glsfmtplural & \Glsxtrheadplural
\Glsxtrtitleplural §5.3.3; 225, 921, see also \Glsfmtplural & \Glsxtrheadplural
\Glsxtrtitleplural §5.3.3; 224, 922, see also \Glsfmtplural & \Glsxtrheadplural
\Glsxtrtitleshort §5.3.3; 218, 922
Index


I

\ifsctstring ............................................. 935
\ifcsundef .............................................. 308, 859
\ifcsvoid .............................................. 308, 530, 593, 756
\ifdefempty ........................................... 308
\ifdefstring ........................................... 859
\ifglsentryexists ..................................... 732, 935, see also
\glsdoifexistsordo, \glsdoifexists & \glsdoifnoexists
\ifglsfieldcseq ....................................... 935
\ifglossaryexists ..................................... 380, 699, 934, see also
\doiglossarynoexistsordo & \glsxtrifemptyglossary
\ifglsentryexists ..................................... 732, 935, see also
\glsdoifexistsordo, \glsdoifexists & \glsdoifnoexists
\ifglsfieldcseq ....................................... 935
\ifglsfielddefeq ....................................... 935
\ifglsfieldeq .......................................... 758, 935
\ifglsfieldvoid ....................................... 308, 498, 935, see also
\GlsXtrIfFieldUndef
\ifglschildren ......................................... 580, 935, see also
\GlsXtrIfHasNonZeroChildCount
\ifglshasdesc ......................................... 480, 936
\ifglshasdescsuppressed ................................ 936
\ifglshasfield 308, 728, 860, 936, see also
\glsxtrifhasfield & \ifglsfieldvoid
\ifglshaslong .......................................... 936
\ifglshasparent ....................................... 295, 580, 936
\ifglshashort .......................................... 256, 936
\ifglshassymbol ....................................... 937
\ifglsindexonlyfirst ................................ 5.8; 268, 937
\ifGlsLongExtraUseTabular 8.7.2; 451, 782, 937
\ifglsnoarrayskip ..................................... 434, 937
\ifglsresetcurrcount 6.1; 311, 791, 937
\ifglsused 15, 247, 268, 284–286, 863, 937, see also
\GlsXtrIfUnusedOrUndefined & \glsxtrifwasfirstuse
\ifglsxtrinitwrglossbefore 5.1.2; 194, 937
\ifglsxtrinsertinside 131, 174, 176, 177, 866, 872, 909, 938
\ifGlsXtrPrefixLabelFallbackLast 11.5.7; 589, 898, 938
\ifglsxtrprintglossflatten 8.4.3; 408, 938
\ifinlistcs ............................................ 308, 841
\ifmglsused ............................................ 7.7; 355, 938
\ifmultiglossaryentryglobal ................................ 7; 328, 938, 957
\IfTeXParserLib §11.5.8; 598, 938
\IfTrackedLanguageFileExists 328, 938
\ifundef .............................................. 308
ignored location (or record) ................................ 265, 538, 547, 549, 627
\IN §11.5.2; 561, 939
\index .................................................. 264, 526, 527, 600–602, 604
indexing application .................................... 195, 264, 265, 314, 431, 626, 627, 987, 989
\indexname .............................................. 939, 990
\indexspace .............................................. 435, 436
inline full form ......................................... 50, 51, 53, 54, 63, 65, 67, 73, 74, 81, 88, 93, 100, 101, 103, 105–107, 109, 111, 114, 132, 170, 175, 253, 285, 625, 627, 662, 665, 833, 847, 865, 866, 901
inner formatting ........................................ 132, 135, 137, 138, 160, 163, 174, 176–178, 236, 240, 242, 244–246,
### Index

<table>
<thead>
<tr>
<th>Page Numbers</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>509, 627, 642, 708, 741, 750, 757, 834, 848, 872, 875, 909, 913</td>
<td>\input \inputenc \inputenc package \inputenc package \inputenc package</td>
</tr>
<tr>
<td>604</td>
<td>\listcsadd \listcsadd \listcsadd \listcsadd</td>
</tr>
<tr>
<td>35, 37, 267, 306, 626, 628</td>
<td>\input \inputenc \inputenc \inputenc \inputenc \inputenc</td>
</tr>
<tr>
<td>34, 628, 633</td>
<td>\interyear \interyear \interyear \interyear \interyear \interyear</td>
</tr>
<tr>
<td>38–40, 135, 231, 245, 246, 294, 296, 298, 304–306, 308, 369, 402, 421, 425, 467, 468, 491, 498, 592, 628, 705, 741, 742, 750, 757, 761, 794, 802, 813, 832, 838, 841, 844, 845, 851, 858–860, 862, 870, 871, 906, 928, 935, 936, 946</td>
<td>\input \inputenc \inputenc \inputenc \inputenc \inputenc</td>
</tr>
<tr>
<td>internal field name</td>
<td>see glossary entry fields</td>
</tr>
<tr>
<td>\INTERPRET</td>
<td>§11.5.2; 561, 939</td>
</tr>
<tr>
<td>\Iota</td>
<td>§11.5.8; 599, 939</td>
</tr>
<tr>
<td>\item</td>
<td>434, 435, 526, 672</td>
</tr>
<tr>
<td>\jobname</td>
<td>17, 535, 536, 870</td>
</tr>
<tr>
<td>\Kappa</td>
<td>§11.5.8; 599, 939</td>
</tr>
<tr>
<td>\LC</td>
<td>§11.5.2; 561, 940</td>
</tr>
<tr>
<td>\leavevmode</td>
<td>32</td>
</tr>
<tr>
<td>\let</td>
<td>§11.5.2; 561, 940</td>
</tr>
<tr>
<td>\letabbreviationstyle</td>
<td>§4.5.3; 161, 940</td>
</tr>
<tr>
<td>\letcs</td>
<td>500</td>
</tr>
<tr>
<td>\levelchar</td>
<td>602</td>
</tr>
<tr>
<td>\linewidth</td>
<td>454, 462</td>
</tr>
<tr>
<td>\loadglsentries</td>
<td>§3.1; 16, 32, 39, 98, 940</td>
</tr>
<tr>
<td>location, ignored/invisible</td>
<td>see ignored location (or record)</td>
</tr>
<tr>
<td>location encap (format)</td>
<td>188, 195, 358, 538, 581, 627, 628, 640, 642, 757, 759, 786, 924</td>
</tr>
<tr>
<td>\longnewglossaryentry</td>
<td>§3.1; 16, 17, 32, 539, 896, 940</td>
</tr>
<tr>
<td>\longnewglossaryentry*</td>
<td>§3.1; 32, 390, 940</td>
</tr>
<tr>
<td>longtable package</td>
<td>482, 677</td>
</tr>
<tr>
<td>lowercase</td>
<td>136, 157, 200, 265, 578, 579, 783, 879</td>
</tr>
<tr>
<td>\mainmatter</td>
<td>188</td>
</tr>
<tr>
<td>\MakeAcronymsAbbreviations</td>
<td>§4.6; 181, 182, 941</td>
</tr>
</tbody>
</table>
Index

\makefirstuc 197–200, 242, 294, 295, 376, 706, 745, 748, 749, 784, 787, 788, 793, 805, 846, 847, 871, 874, 908, 912, 941, 942, 943
\makeglossaries §8; 23–25, 258, 378, 386, 389, 614, 789, 889, 941, 968, 969, 987, 992
\makeglossaries ............. 6, 258, 315, 316
\makeindex 6, 17, 20, 23, 25, 28, 195, 196, 264, 600, 627, 941, 968, 987, 990
\makenoidxglossaries 16, 17, 24, 25, 378, 386, 402, 538, 581, 789, 889, 941, 969, 987, 992
\MakeTextLowercase ............. 200
\MakeTextUppercase ............. 203, 227, 941
\MakeUppercase ............. 149, 150, 152, 227
\markboth 201, 204, 214, 227, 423, 680, 861, 877, 901
\markright 201, 204, 214, 227, 423, 680, 861, 877, 901
\mbox ................................ 5
\memoir class .................. 242, 243, 287, 291, 292
\mfirstuc package 197–200, 203, 228, 235, 244, 294, 302, 525, 537, 692, 693, 783, 784, 882, 941–943, 981, 990
\mfirstucMakeUppercase ............. 200, 376, 941
\MFUaddmap . 199, 200, 302, 784, 942, see also \MFUexcl & \MFUblocker
\MFUblocker ............. 199, 784, 942, see also \MFUexcl & \MFUaddmap
\MFUexcl ............. 198, 784, 942, see also \MFUblocker & \MFUaddmap
\MFUsave ............. 537, 882, 942
\MFUsaveatend ............. 538, 942
\MFUsentencecase . 197, 198, 242, 302, 490, 784, 891, 941, 942
\MGP ................................ $11.5.2; 560, 561, 727, 943
\MGLS ................................ $7.11.1; 345, 360, 367, 943
\MGLs ................................ $7.11.1; 366, 943
\Mgl ................................ $7.11.1; 343, 345, 360, 366, 943
\mglAddOptions ............. $7; 328, 943
\mglcurrentcategory §7.5; 351, 357, 944
\mglcurrentlabel ............. $7.5; 351, 361, 944
\mglcurrentlist ............. $7.5; 350, 944
\mglcurrentmainlabel ............. $7.5; 350, 944
\mglcurrentmultilabel ............. $7.5; 350, 944
\mglcurrentoptions ............. $7.5; 350, 944
\mglcurrentprefix ............. $7.5; 344, 345, 351, 944
\mglcurrentsuffix ............. $7.5; 345, 351, 945
\mglcustompostlinkhook ............. §7.6; 352, 360, 945
\mgldefcategoryprefix ............. §7.3; 344, 945
\mgldefcategorysuffix ............. §7.3; 344, 945
\mglelementindex ............. $7.5; 351, 945
\mglelementposthook ............. $7.5; 350, 945
\mglelementprehook ............. $7.5; 350, 351, 369, 945
\mglelementreset ............. §7.10; 365, 945
\mglelementunset ............. §7.10; 365, 946
\mgsfield ............. §7.11.3; 369, 946, 955
\mgsforelements ............. §7.13; 373, 374, 946
\mgsfortherelements ............. §7.13; 374, 946
\Mgsfull ............. §7.11.2; 368, 946
\Mgsfull ............. §7.11.2; 367, 368, 946
\mgsheascategoryprefix ............. §7.3; 345, 947
\mgsheascategorysuffix ............. §7.3; 345, 947
\mgsiflast ............. $7.5; 352, 947
\mgsiflastelementcasefirstuse ............. §7.6.1; 354, 947
\mgsiflastelementsfirstuse ............. §7.6.1; 353, 947
\mgsiflastelementwasfirstuse ............. §7.6.1; 353, 947
\mgsiflastelementwasplural ............. §7.6.1; 354, 947
\mgsiflastmaincasefirstuse ............. §7.6.2; 355, 948
\mgsiflastmainskipped ............. §7.6.2; 354, 948
\mgsiflastmainwasfirstuse ............. §7.6.2; 354, 948
\mgsiflastmainwasplural ............. §7.6.2; 355, 948
\mgsifirstuse ............. §7.5; 351, 948
\mglslastcategory ............. §7.6; 353, 948

1037
Index

\mglslastelementlabel §7.6.1; 344, 353, 354, 949
\mglslastelementpostlinkhook §7.6; 352, 360, 949
\mglslastmainlabel §7.6.2; 354, 949
\mglslastmainpostlinkhook §7.6; 352, 360, 949
\mglslastmultilabel §7.6; 353, 949
\mgslocalreset §7.7; 356, 949
\mgslocalunset §7.7; 356, 949
\mgslocalunsetothers §7.10; 365, 950
\Mglslong §7.11.2; 368, 950
\mgslong §7.11.2; 367, 368, 950
\MGLSmainpl §7.11.1; 367, 950
\MGLsmainpl §7.11.1; 367, 950
\Mgsmainpl §7.11.1; 366, 950
\mgsmainpl §7.11.1; 343, 349, 366, 951
\MGLsname §7.11.3; 368, 951
\mglsname §7.11.3; 349, 355, 368, 951
\MGLSpl §7.11.1; 367, 951
\MGlspl §7.11.1; 367, 950
\Mgspl §7.11.1; 366, 951
\mgspl §7.11.1; 343, 349, 363, 952
\mgsprefix §7.3; 343, 345, 952
\mgsreset §7.7; 355, 952
\mgsresetall §7.7; 356, 952
\mgsseefirstitem §7.12; 304, 372, 792, 952
\mgsseeitem §7.12; 304, 372, 792, 952
\mgsSetMain §7.12; 327, 342, 952
\mgsSetOptions §7; 328, 953
\Mglsshort §7.11.2; 368, 953
\mgsshort §7.11.2; 367, 368, 953
\mgsuffix §7.3; 344, 345, 353, 953
\MGLssymbol §7.11.3; 368, 953
\Mglssymbol §7.11.3; 369, 953
\mglssymbol §7.11.3; 368, 954
\mglsunset §7.7; 355, 954
\mglsunsetall §7.7; 356, 954
\mgsunsetothers §7.10; 365, 954
\mgsusecategoryprefix §7.3; 345, 954
\mgsusecategoriesuffix §7.3; 345, 954
\MGlssufﬁeld §7.11.3; 369, 954
\Mglsusefield §7.11.3; 369, 946, 954, 955
\mglswasfirstuse §7.6; 353, 955
\mini-glossary 293, 412, 416, 464, 557, 558
\MPGLS §7.11.4; 372, 955
\MPGLsp §7.11.4; 371, 955
\MPgl §7.11.4; 371, 955
\mpgls §7.11.4; 371, 955
\mpgls §7.11.4; 343, 360, 370, 955, 957
\mpglsMainpl §7.11.4; 372, 956
\MPGLsp §7.11.4; 371, 956
\MPGLSp §7.11.4; 372, 956
\mpglspl §7.11.4; 371, 956
\mpglspl §7.11.4; 371, 956
\mpglsWarning §7.11.4; 370, 957
\Mu §11.5.8; 599, 957
multi-entry first use 359–362, 626, 628, 645–647
multi-entry first use flag 364–366, 628, 647
multi-entry set options 645
all §7.10; 362–364, 645
category §7.9.6; 356, 362, 645
capmain §7.9.2; 358, 645
capother §7.9.2; 358, 645
firstprefix §7.9.4; 345, 360, 645
firstskipmain §7.9.5; 361, 645
firstskipother §7.9.5; 361, 646
firstsuffix §7.9.4; 331, 345, 360, 646
hyper §7.10; 330, 335, 364, 646
hyper §7.9.6; 361, 646
indexmain §7.9.1; 358, 646
indexother §7.9.1; 328, 358, 646
main §7.10; 363, 646
mglsopts §7.9.6; 335, 362, 636, 647
mpostlink §7.9.3; 333, 352, 359, 647
mpostlinkelement §7.9.3; 352, 359, 647
multiunset §7.10; 364, 366, 647
others §7.10; 335, 363, 647
postlinks §7.9.3; 333, 350, 352, 359, 647
presetlocal §7.10; 335, 364, 646, 647
1038
Index

resetall  §7.10; 364, 647, 946
resetmain  §7.10; 364, 648
resetothers  §7.10; 364, 648
setup  §7.10; 357, 358, 362, 363, 648
textformat  §7.9.6; 361, 362, 648
unsetall  §7.10; 365, 648, 946
unsetmain  §7.10; 365, 648
unsetothers  §7.10; 365, 648
usedprefix  §7.9.4; 345, 360, 649
usedskipmain  §7.9.5; 361, 649
usedskipothers  §7.9.5; 361, 649
usedsuffix  §7.9.4; 345, 360, 649
multi-entry subsequent use  359–361, 628, 647, 649
multicol package  475
multicols environment  444, 476, 677, 678, 806, 828, 830
multicols* environment  §7; 326–328, 342, 356, 357, 363, 374, 375, 957, 973
multiglossaryentryglobalfalse  §7; 328, 957
multiglossaryentryglobaltrue  §7; 327, 957
multiglossaryentrysetup  §7.9; 350, 357, 363, 957

\nameref  215
\nameref package  215
\newabbr  §4.3.2; Table 4.1; 958
\newabbreviation  §4.1; Table 4.1; 5, 16, 41, 43–45, 47, 50, 63, 66, 82, 94, 115, 133, 161, 163–167, 170, 181, 185, 326, 380, 390, 500, 517, 518, 521–523, 539, 631–637, 654, 681, 685–687, 700, 727, 729, 745, 748, 749, 761, 783, 787, 788, 795, 805, 886, 890, 940, 958, 976, 987
\newabbreviationhook  §4.1.5; 45, 162, 958
\newabbreviationstyle  §4.5.3; 43, 159–162, 170, 958
\newacronym  3, 5, 7, 10, 19, 42–44, 59, 60, 181, 518, 521–523, 635, 654, 958, 974, 976, 988
\newglossary*  §2.4; 18, 959, 988
\newignoredglossary  257, 378, 379, 627, 960, 973
\newignoredglossary*  §8; 378, 528, 960
\newnum  §2.4; 19, 960, 988
\newsym  §2.4; 18, 960, 988
\newterm  §2.1; 12, 519, 960, 990
\NIN  §11.5.2; 561, 961
\NoCaseChange  203, 216
\nonbreaking  332, 348, 794
\nopostdesc  12, 32, 421, 428, 821, 936, 961, see also \glsxtrnopostpunct
\NOTPREFIXOF  §11.5.2; 561, 961
\NOTSUFFIXOF  §11.5.2; 561, 961
\Nu  §11.5.8; 599, 961
\NULL  §11.5.2; 561, 961
\null  6
\numberlist  see location list
\Omicron  §11.5.8; 599, 962
\omicron  §11.5.8; 599, 962

\pagelistname  451, 962
\pageref  380, 891
\pagetitle  206
\par  786
\parindent  479
\part  265
\partcounter  265
\pdfbookmark  448, 828

1039
Index

\PGLS . 372, 962
\Pgl . 55, 207, 371, 962
\pgl . 55, 370, 371, 956, 962, 964, 965
\PGLSfmtlong . §5.3.2; 209, 962
\pglsfmtlong . §5.3.2; 209, 219, 963, 967
\PGLSfmtlongpl . §5.3.2; 209, 962, 963
\pglsfmtlongpl . §5.3.2; 209, 219, 963, 967
\PGLSfmtshort . §5.3.2; 207, 963
\pglsfmtshort . §5.3.2; 207, 963
\PGLSpl . 372, 964
\pglspl . 371, 965
\PGLSxtrlong . §4.3.1; 56, 965
\pglsxtrlong . §4.3.1; 56, 965
\pglsxtrlongpl . §4.3.1; 56, 966
\PGLSxtrlongpl . §4.3.1; 56, 966
\PGlsxtrshort . §4.3.1; 55, 966
\PGLSxtrshort . §4.3.1; 55, 966
\pglsxtrshort . §4.3.1; 55, 966
\pglsxtrshortpl . §4.3.1; 55, 966
\PGLSxtrshortpl . §4.3.1; 55, 967
\pglsxtrshortpl . §4.3.1; 55, 967
\PGLSxtrtitlelong . §5.3.3; 219, 967
\pglsxtrtitlelong . §5.3.3; 219, 967
\PGLSxtrtitlelongpl . §5.3.3; 219, 967
\pglsxtrtitlelongpl . §5.3.3; 219, 967
\PGLSxtrtitleshort . §5.3.3; 219, 967
\pglsxtrtitleshort . §5.3.3; 219, 967
polyglossia package . 618
post-name hook . 407, 419, 425, 426, 452, 465, 526, 625, 629, 630, 634, 730, 897
\PREFIXOF . §11.5.2; 561, 968
\preto . 165, 396, 404
\pretoglossarypreamble . §8.2; 381, 968
print [unsrt|noidx] glossary options . 381, 650
entrycounter . 650
flatten . §8.3; 383, 395, 405, 406, 408, 627, 650, 729, 924, 938
groups . §8.3; 383, 389, 406, 488, 650, 756
label . §8.3; 382, 489, 650
leveloffset . §8.3; 382, 383, 405, 406, 486, 627, 650, 728, 729, 924
nogroupskip . 389, 434, 488, 651
nonumberlist . 35, 488, 628, 651, 990
nopostdot . 651
numberedsection . §8.3; 382, 651
postamble . §8.3; 386, 651
preamble . §8.3; 386, 651
prefix . §8.3; 257, 383, 385, 652
sort . §8.3; 381, 652
style . §8.3; 382, 483, 488, 652, 671
subentrycounter . 652
target . §8.3; 257, 385, 652
targetnameprefix . §8.3; 384, 385, 415, 652
title . §8.3; 379, 381, 409, 483, 489, 618, 652, 959
tocitle . §8.3; 381, 483, 489, 653
type . 391, 409, 416, 653, 968–970, 972, 973
\printabbreviations . §2.1; 9, 968, 984
\printacronyms
\printglossaries
\printglossary
\printindex
\printnoidxglossaries
\printnoidxglossary
\printnumbers
\printsymbols
\printunsrtabbreviations
\printunsrtacronyms
\printunsrtglossaries
\printunsrtglossary
\printunsrtglossary*
\printunsrtglossaryentryprocesshook
\printunsrtglossarygrouphook
\printunsrtglossaryhandler
\printunsrtglossarypostbegin
\printunsrtglossarypostentryprocesshook
\printunsrtglossarypostskip
\printunsrtglossarysetup
\printunsrtglossaryunitskip
\printunsrtglossarywrapenvironment
\printunsrtindex
\printunsrtinnerglossary
\printunsrtnumbers
\printunsrtsymbols
\printunsrttable
\printunsrtacronym
\printunsrtglossary
\printunsrtglossaryunitpostskip
\printunsrtglossaryunitsetup
\protect
\providecommand
\block-style
\desc-name
\desc-other-name
\desc-other-symbol-name
\desc-symbol-other-name
\name-desc-symbol
\name-desc
\name-other-desc
\name-other-symbol-desc
\name-other
\name-symbol-desc
\name-symbol-other-desc
\name-symbol
\name
\other-name
\other
\par
\rules
\protect
\providecommand
Index

695, 699, 700, 899, 939, 957, 961, 962, 975, 978–982
\provideignoredglossary \§8; 379, 580, 973
\provideglossaryentry \§7; 327, 375, 973
\ProvidesGlossariesExtraLang \§15; 619, 973

\quotechar \...... 602

\ref \...... 380, 790, 891
\refstepcounter \...... 25, 27, 320, 582
\relax \...... 280, 294, 332, 530, 592, 736, 822, 903, 928
\relsize package \...... 63, 157, 658, 660, 662, 669, 670
\renewabbreviationstyle \§4.5.3; 161, 973, see also
\newabbreviationstyle
\renewcommand \...... 251, 428, 598, 599, 791, 899
\RequireGlossariesExtraLang 973
\resource file \...... 598, 629
\resource options \...... 24–26, 375, 425, 560, 562, 580, 587, 589, 629, 651, 725, 820, 834, 978
\alias-loc \...... 271, 281, 282
\alias \...... 281
\assign-fields \...... 537, 560, 561, 901
\category \...... 553
\combine-dual-locations \...... 596, 597
\compound-adjust-name \...... 375, 883
\description-case-change \...... 525
\dual-backlink \...... 589
\dual-field \...... 587, 588
\dual-prefix \...... 594
\dual-type \...... 587
\entry-type-aliases \...... 562, 827
\field-aliases \...... 562
\flatten \...... 383
\group-level \...... 395, 401, 479, 480
\group \...... 544
\interpret-fields \...... 425
\label-prefix \...... 594
\loc-counters \...... 554
\loc-prefix \...... 430, 433
\loc-suffix \...... 430
\name-case-change \...... 526
\post-description-dot \...... 2, 12, 13
\prune-xr \...... 278
\replicate-fields \...... 425
\save-child-count \...... 483, 580, 640, 860, 936
\save-index-counter \...... 27, 599, 640
\save-locations \...... 25, 35, 446, 475, 538, 628, 633, 651, 987, 991
\secondary \...... 383, 394, 395
\see \...... 281, 447
\seealso \...... 281, 447
\selection \...... 22, 23, 261, 269, 271, 277, 281, 282, 539, 549, 586, 725
\set-widest \...... 440, 452, 463, 477, 579, 908
\sort-field \...... 34
\sort-label-list \...... 302
\sort-replace \...... 560
\sort-rule \...... 546, 563, 564
\sort \...... 25, 381, 539, 563, 987
\src \...... 17, 539, 580
\supplemental-locations \...... 195, 581, 586
\symbol-sort-fallback \...... 415
\trigger-type \...... 550, 555
\type \...... 543
\resource set \...... 580, 629, 820, 900
\restoreacronyms \§4.6; 181, 182, 941, 974
\rGLS \§11.4; 199, 260, 551, 552, 887, 974, see also \glsxtrifrecordtrigger,
\rGLSformat & --record-count
\rGls \§11.4; 200, 260, 551, 552, 887, 974, see also \glsxtrifrecordtrigger,
\rGlsformat & --record-count
\rgls \§11.4; 56, 183, 200, 260, 549, 550, 552, 556, 558, 839, 887, 974, 975, see also \glsxtrifrecordtrigger,
\rglsformat & --record-count

1042
\rGLSformat \textsection11.4; 551, 974
\rGlsformat $\textsection11.4; 551, 974$
\rglsformat $\textsection11.4; 550, 553, 974$
\rGLSpl $\textsection11.4; 260, 551, 552, 887, 975$, see also \glsxtrifrecordtrigger,
\rGLSplformat & --record-count
\rGlsplformat $\textsection11.4; 260, 551, 552, 887, 975$, see also \glsxtrifrecordtrigger,
\rglsplformat & --record-count
\rglsplformat $\textsection11.4; 551, 975$
\rglsplformat $\textsection11.4; 551, 975$
\rglsplformat $\textsection11.4; 550, 975$
\Rho \textsection11.5.8; 599, 975
\Roman \textsection265
\roman \textsection265, 266

S

\sample{sample}{\textsection14.2; 608}
\sample{sample-abbr-styles}{\textsection14.2; 608}
\sample{sample-abbrev}{\textsection14.2; 609}
\sample{sample-accsupp}{\textsection14.2; 616}
\sample{sample-acronym}{\textsection14.2; 609}
\sample{sample-acronym-desc}{\textsection14.2; 610}
\sample{sample-alias}{\textsection14.2; 614}
\sample{sample-altmodifier}{\textsection14.2; 613}
\sample{sample-alttree}{\textsection14.2; 443, 614}
\sample{sample-alttree-marginpar}{\textsection14.2; 443, 615}
\sample{sample-alttree-sym}{\textsection14.2; 443, 615}
\sample{sample-autoindex}{\textsection14.2; 612}
\sample{sample-autoindex-hyp}{\textsection14.2; 612}
\sample{sample-crossref}{\textsection14.2; 610}
\sample{sample-entrycount}{\textsection14.2; 612}
\sample{sample-external}{\textsection14.2; 527, 614}
\sample{sample-fmt}{\textsection14.2; 614}
\sample{sample-footnote}{\textsection14.2; 610}
\sample{sample-header}{\textsection14.2; 611}
\sample{sample-indexhook}{\textsection14.2; 610}
\sample{sample-initialisms}{\textsection14.2; 611}
\sample{sample-linkcount}{\textsection14.2; 613}
\sample{sample-mixed-abbrev-styles}{\textsection14.2; 611}
\sample{sample-mixedsort}{\textsection14.2; 378, 614}
\sample{sample-mixture}{\textsection14.2; 609}
\sample{sample-name-font}{\textsection14.2; 609}
\sample{sample-nested}{\textsection14.2; 612}
\sample{sample-onelink}{\textsection14.2; 613}
\sample{sample-onethefly}{\textsection14.2; 615}
\sample{sample-onethefly-utf8}{\textsection14.2; 615}
\sample{sample-onethefly-xetex}{\textsection14.2; 615}
\sample{sample-pages}{\textsection14.2; 429, 613}
\sample{sample-postdot}{\textsection14.2; 611}
\sample{sample-postlink}{\textsection14.2; 611}
\sample{sample-prefix}{\textsection14.2; 616}
\sample{sample-suppl}{\textsection14.2; 617}
\sample{sample-suppl-hyp}{\textsection14.2; 617}
\sample{sample-suppl-main}{\textsection14.2; 616}
\sample{sample-suppl-main-hyp}{\textsection14.2; 616}
\sample{sample-undef}{\textsection14.2; 610}
\sample{sample-unitentrycount}{\textsection14.2; 613}
\section \textsection704
\sectioncounter \textsection316, 548, 581, 583, 584
\seealso \textsection33, 270, 281, 631, 635, 864, 976
\seename \textsection270, 635, 791, 976
\sentencecase \textsection254, 525, 706, 727, 745, 748, 787, 788, 805, 846, 847, 871, 874, 908, 912, 941, 943, 947, 948
\setabbreviationstyle \textsection4.5; 7, 41, 59, 170, 958, 976
\setacronymstyle \textsection43, 229, 976
\setentrycounter \textsection582, 976
\setglossarypreamble \textsection381, 976
\setglossarystyle \textsection398, 424, 483, 671, 898, 977
\setupglossaries \textsection2; 9, 434, 977
\setupglsadd \textsection5.1.1; 190, 977, see also \setupglslink
\setupglslink \textsection5.1.1; 190, 977, see also \setupglsadd

1043
Index

\so ........................................ 243, 244
soul package ............................ 237, 242, 243, 287
standalone entry ........................ 629, 960
\stepcounter ................................ 320, 864
\string ................................. 537, 546, 560, 561, 565, 567, 601, 727, 757, 978
\subglossentry .......................... 402, 403, 405, 419, 483, 977
subsequent use .......................... 630
\SUFFIXOF ................................ §11.5.2; 561, 977
supertabular environment .......... 678
\symbolname ............................. 451, 490, 499, 978

T
\tabcolsep 454, 458, 459, 462, 472, 473, 496
table environment .................... 27
\tableofcontents ........................ 203, 215
tabular environment ................. 387, 404, 407, 434, 450, 451, 483, 671, 673–677, 696, 700, 731, 782, 786, 937, 962, 978
\tabularnewline .......................... 406, 494
tabulation ( &) ......................... 406, 483
\Tau ........................................ §11.5.8; 599, 978
texindy .................................... 600
\texorpdfstring .......................... 205, 216, 424
\textbf .................................. 436, 499, 586
textcase package ...................... 203
\textlarger ............................... 158, 914
\textsc 61, 136, 156, 157, 658–660, 662, 667, 688, 805, 842, 902
\textsmaller 63, 156, 157, 230, 658, 660, 662, 669, 670, 842, 913
\texts subscript ........................ 500
\textsuperscript ........................ 500
\textup .................................... 158, 838
\the(counter) .......................... 196, 316, 321, 413, 414, 547, 554, 582
theindex environment .............. 602
\thepage .................................. 547, 549
\TITLE .................................... §11.5.2; 561, 978
title case 200, 294, 375–377, 525, 693, 739, 842, 883
tracklang package ........................ 296, 297, 539, 564, 566, 619, 620, 845
translator package ................... §11.5.2; 561, 978
\TRIM .................................... 3, 993
\u ........................................... §11.5.2; 537, 546, 560, 561, 757, 978, see also \glsxtr\Resource\InitEscSequences
\UC ....................................... §11.5.2; 561, 978
\ul ......................................... 237
\underline .................................. 58
\unskip .................................... 20, 32, 282
\Upalpha .................................. §11.5.8; 579, 599, 979
\upalpha .................................. 577–579
\Upbeta .................................. §11.5.8; 599, 979
\Upchi .................................. §11.5.8; 599, 979
\Upsilon .................................. §11.5.8; 599, 979
\Upmu .................................. §11.5.8; 599, 980
\Upnu .................................. §11.5.8; 599, 980
\Upomicron ................................ §11.5.8; 599, 980
\upgreek package ........................ 577, 578, 599, 979–981
\Upiota .................................. §11.5.8; 599, 979
\Upkappa .................................. §11.5.8; 599, 980
\Up epsilon .............................. §11.5.8; 599, 979
\Upeta .................................. §11.5.8; 599, 979
upgreek package ....................... 577, 578, 599, 979–981
\Upiota .................................. §11.5.8; 599, 979
\Upkappa .................................. §11.5.8; 599, 980
\Up eta .................................. §11.5.8; 599, 979
\Up mu .................................. §11.5.8; 599, 980
\Up nu .................................. §11.5.8; 599, 980
\Upomicron ................................ §11.5.8; 599, 980
\upmicron ................................ §11.5.8; 599, 980
\uppercase ................................ 228
upercase 136, 138, 146, 157, 197, 200, 201, 265, 294–296, 303, 314, 392, 393, 525, 578, 579, 706, 734, 745, 748, 787, 788, 805, 814, 846, 847, 856, 871, 874, 881, 908, 912, 941, 943, see also title case, sentence case & all caps
\Up rho .................................. §11.5.8; 599, 980
\Uptau .................................. §11.5.8; 599, 980
\Up zeta .................................. §11.5.8; 599, 981
UTF-8 184, 197, 198, 250, 432, 443, 546, 604, 630, 822, 907

1044
Index

W

whatsit ................... 185, 194, 267, 293, 630
wrglossary counter §2.4; 27, 29, 584, 585, 640, 934, 985, 986
\writemultiglossentry §7.13; 374, 981

X

\xcapitalisefmtwords ............ 525, 981
xfor package .................... 305, 374
\xglissetwidest ........... §8.6.5.4; 439, 981
\xglsupdatewidest ........... §8.6.5.4; 440, 981
\xGlsXtrIfValueInFieldCsvList §5.13; 307, 981
\xGlsXtrSetField ............ §3.5; 39, 40, 982
\xifinlistcs .................... 308, 842
\xindex ........................ 627
\xindy 6, 17, 20, 23, 25, 28, 195, 196, 264, 600, 627, 636, 941, 968, 987, 993
\xpglsxtrpostabbrvfootnote §4.5.2; 160, 982
\xpglsxtrposthyphenlong ........ 152, 982
\xpglsxtrposthyphenshort .. 150, 982
\xpglsxtrposthyphensubsequent 151, 982

Z

\Zeta ......................... §11.5.8; 599, 982